Multimedia Data-Embedding and Watermarking Technologies

MITCHELL D. SWANSON, MEMBER, IEEE, MEI KOBAYASHI, AND AHMED H. TEWFIK, FELLOW, IEEE

Invited Paper

In this paper, we review recent developments in transparent data embedding and watermarking for audio, image, and video. Data-embedding and watermarking algorithms embed text, binary streams, audio, image, or video in a host audio, image, or video signal. The embedded data are perceptually inaudible or invisible to maintain the quality of the source data. The embedded data can add features to the host multimedia signal, e.g., multilingual soundtracks in a movie, or provide copyright protection. We discuss the reliability of data-embedding procedures and their ability to deliver new services such as viewing a movie in a given rated version from a single multicast stream. We also discuss the issues and problems associated with copy and copyright protections and assess the viability of current watermarking algorithms as a means for protecting copyrighted data.

Keywords—*Copyright protection, data embedding, steganography, watermarking.*

I. INTRODUCTION

The past few years have seen an explosion in the use of digital media. Industry is making significant investments to deliver digital audio, image, and video information to consumers and customers. A new infrastructure of digital audio, image, and video recorders and players, on-line services, and electronic commerce is rapidly being deployed. At the same time, major corporations are converting their audio, image, and video archives to an electronic form.

Manuscript received July 15, 1997; revised January 15, 1998. The Guest Editor coordinating the review of this paper and approving it for publication was A. M. Tekalp. This work was supported in part by the Air Force Office of Scientific Research under Grant AF/F49620-94-1-0461 and in part by the Advanced Research Project Agency under Grant AF/F49620-93-1-0558.

M. D. Swanson is with Cognicity, Inc., Minneapolis, MN 55344 USA (e-mail: swanson@cognicity.com).

M. Kobayashi is with the Graduate School of Mathematical Sciences, University of Tokyo and IBM Tokyo Research Laboratory, Yamato-shi, Kanagawa-ken 242 Japan (e-mail: mei@trl.ibm.co.jp).

A. H. Tewfik is with Cognicity, Inc., Minneapolis, MN 55344 USA (e-mail: tewfik@cognicity.com) and the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: tewfik@ece.umn.edu).

Publisher Item Identifier S 0018-9219(98)03519-1.

Digital media offer several distinct advantages over analog media: the quality of digital audio, image, and video signals is higher than that of their analog counterparts. Editing is easy because one can access the exact discrete locations that should be changed. Copying is simple with no loss of fidelity. A copy of a digital media is identical to the original. Digital audio, image, and videos are easily transmitted over networked information systems.

These advantages have opened up many new possibilities. In particular, it is possible to hide data (information) within digital audio, image, and video files. The information is hidden in the sense that it is perceptually and statistically undetectable. With many schemes, the hidden information can still be recovered if the host signal is compressed, edited, or converted from digital to analog format and back.

As we shall see in Section II, pure analog data-hiding techniques had been developed in the past. However, these techniques are not as robust as most of the digital data hiding techniques that we review in this paper. Furthermore, they cannot embed as much data in a host signal as the digital approaches.

Digital data embedding has many applications. Foremost is passive and active copyright protection. Many of the inherent advantages of digital signals increase problems associated with copyright enforcement. For this reason, creators and distributors of digital data are hesitant to provide access to their intellectual property. Digital watermarking has been proposed as a means to identify the owner or distributor of digital data.

Data embedding also provides a mechanism for embedding important control, descriptive, or reference information in a given signal. This information can be used for tracking the use of a particular clip, e.g., for pay-per-use applications, including billing for commercials and video and audio broadcast, as well as Internet electronic commerce of digital media. It can be used to track audio or visual object creation, manipulation, and modification history within a given signal without the overhead associated with creating

E- 6

he knows that the host signal contains data and is familiar with the exact algorithm for embedding the data. Note that in some applications, e.g., covert communications, the data may also be encrypted prior to insertion in a host signal.

F. Copyright Protection and Ownership Deadlock

Data-embedding algorithms may be used to establish ownership and distribution of data. In fact, this is the application of data embedding or watermarking that has received most attention in the literature. Unfortunately, most current watermarking schemes are unable to resolve rightful ownership of digital data when multiple ownership claims are made, i.e., when a deadlock problem arises. The inability of many data-embedding algorithms to deal with deadlock, first described by Craver *et al.* [15], is independent of how the watermark is inserted in the multimedia data or how robust it is to various types of modifications.

Today, no scheme can unambiguously determine ownership of a given multimedia signal if it does not use an original or other copy in the detection process to at least construct the watermark to be detected. A pirate can simply add his watermark to the watermarked data or counterfeit a watermark that correlates well or is detected in the contested signal. Current data-embedding schemes used as copyright-protection algorithms are unable to establish who watermarked the data first. Furthermore, none of the current data-embedding schemes has been proven to be immune to counterfeiting watermarks that will correlate well with a given signal as long as the watermark is not restricted to depend partially in a noninvertible manner on the signal.

If the detection scheme can make use of the original to construct the watermark, then it may be possible to establish unambiguous ownership of the data regardless of whether the detection scheme subtracts the original from the signal under consideration prior to watermark detection or not. Specifically, [16] derives a set of sufficient conditions that watermarks and watermarking schemes must satisfy to provide unambiguous proof of ownership. For example, one can use watermarks derived from pseudorandom sequences that depend on the signal and the author. Reference [16] establishes that this will work for *all* watermarking procedures regardless of whether they subtract the original from the signal under consideration prior to watermark detection or not. Reference [85] independently derived a similar result for a restricted class of watermarking techniques that rely on subtracting a signal derived from the original from the signal under consideration prior to watermark detection. The signal-dependent key also helps to thwart the "mix-and-match" attack described in [16].

An author can construct a watermark that depends on the signal and the author and provides unambiguous proof of ownership as follows. The author has two random keys x_1 and x_2 (i.e., seeds) from which a pseudorandom sequence y can be generated using a suitable pseudorandom sequence generator [76]. Popular generators include RSA, Rabin, Blum/Micali, and Blum/Blum/Shub [25]. With the two proper keys, the watermark may be extracted. Without the two keys, the data hidden in the signal are statistically undetectable and impossible to recover. Note that classical maximal length pseudonoise sequences (i.e., m-sequence) generated by linear feedback shift registers are *not* used to generate a watermark. Sequences generated by shift registers are cryptographically insecure: one can solve for the feedback pattern (i.e., the keys) given a small number of output bits y.

The noise-like sequence y may be used to derive the actual watermark hidden into the signal or to control the operation of the watermarking algorithm, e.g., to determine the location of pixels that may be modified. The key x_1 is *author* dependent. The key x_2 is *signal* dependent. The key x_1 is the secret key assigned to (or chosen by) the author. The key x_2 is *computed from the signal* that the author wishes to watermark. It is computed from the signal using a one-way hash function. For example, the tolerable error levels supplied by masking models (see Section IV) are hashed in [85] to a key x_2. Any one of a number of well-known secure one-way hash functions may be used to compute x_2, including RSA, MD4 [77], and SHA [60]. For example, the Blum/Blum/Shub pseudorandom generator uses the one-way function $y = g_n(x) = x^2 \bmod n$, where $n = pq$ for primes p and q so that $p = q = 3 \bmod 4$. It can be shown that generating x or y from partial knowledge of y is *computationally infeasible* for the Blum/Blum/Shub generator.

The signal-dependent key x_2 makes counterfeiting very difficult. The pirate can only provide key x_1 to the arbitrator. Key x_2 is automatically computed by the watermarking algorithm from the original signal. As it is computationally infeasible to invert the one-way hash function, the pirate is unable to fabricate a counterfeit original that generates a desired or predetermined watermark.

Deadlock may also be resolved using the dual watermarking scheme of [85]. That scheme employs a *pair* of watermarks. One watermarking procedure requires the original data set for watermark detection. The second watermarking procedure does *not* require the original data set. A data-embedding technique that satisfies the restrictions outlined in [16] can be used to insert the second watermark.

The above discussion clearly highlights the limitation of watermarking as an unambiguous mean of establishing ownership. Future clever attacks may show that the schemes described in [16] or [85] are still vulnerable to deadlock. Furthermore, all parties would need to use watermarking techniques that have been proven or certified to be immune to deadlock to establish ownership of media. Note also that contentions of ownership can occur in too many different forms. Copyright protection will probably not be resolved exclusively by one group or even the entire technical community since it involves too many legal issues, including the very definition of similarity and derived works. Many multidisciplinary efforts are currently investigating standards and rules for national and international copyright protection and enforcement in the digital age.

F-7

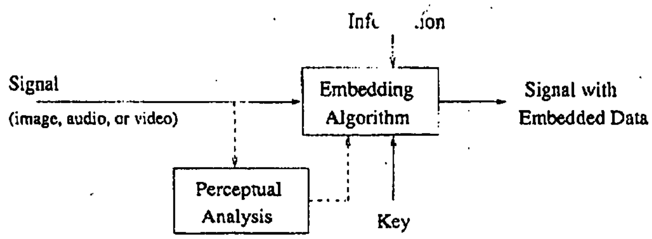


Fig. 1. Diagram of a data-embedding algorithm. The information is embedded into the signal using the embedding algorithm and a key. The dashed lines indicate that the algorithm may directly exploit perceptual analysis to embed information.

IV. Signal Insertion: The Role of Masking

The first problem that all data-embedding and watermarking schemes need to address is that of inserting data in the digital signal without deteriorating its perceptual quality. Of course, we must be able to retrieve the data from the edited host signal, i.e., the insertion method must also be invertible. Since the data-insertion and data-recovery procedures are intimately related, the insertion scheme must take into account the requirement of the data-embedding application. In many applications, we will need to be able to retrieve the data even when the host signal has undergone modifications, such as compression, editing, or translation between formats, including A/D and D/A conversions.

Data insertion is possible because the digital medium is ultimately consumed by a human. The human hearing and visual systems are imperfect detectors. Audio and visual signals must have a minimum intensity or contrast level before they can be detected by a human. These minimum levels depend on the spatial, temporal, and frequency characteristics of the human auditory and visual systems. Further, the human hearing and visual systems are characterized by an important phenomenon called masking. Masking refers to the fact that a component in a given audio or visual signal may become imperceptible in the presence of another signal called the masker. Most signal-coding techniques (e.g., [41]) exploit the characteristics of the human auditory and visual systems directly or indirectly. Likewise, all data-embedding techniques exploit the characteristics of the human auditory and visual systems implicitly or explicitly (see Fig. 1). In fact, embedding data would not be possible without the limitations of the human visual and auditory systems. For example, it is not possible to modify a binary stream that represents programs or numbers that will be interpreted by a computer. The modification would directly and adversely affect the output of the computer.

A. The Human Auditory System (HAS)

Audio masking is the effect by which a faint but audible sound becomes inaudible in the presence of another louder audible sound, i.e., the masker [42]. The masking effect depends on the spectral and temporal characteristics of both the masked signal and the masker.

Frequency masking refers to masking between frequency components in the audio signal. If two signals that occur simultaneously are close together in frequency, the stronger masking signal may make the weaker signal inaudible. The masking threshold of a masker depends on the frequency, sound pressure level, and tone-like or noise-like characteristics of both the masker and the masked signal [61]. It is easier for a broad-band noise to mask a tonal signal than for a tonal signal to mask out a broad-band noise. Moreover, higher frequency signals are more easily masked.

The human ear acts as a frequency analyzer and can detect sounds with frequencies that vary from 10 to 20 000 Hz. The HAS can be modeled by a set of bandpass filters with bandwidths that increase with increasing frequency. The bands are known as the critical bands. The critical bands are defined around a center frequency in which the noise bandwidth is increased until there is a just noticeable difference in the tone at the center frequency. Thus, if a faint tone lies in the critical band of a louder tone, the faint tone will not be perceptible.

Frequency-masking models are readily obtained from the current generation of high-quality audio codecs, e.g., the masking model defined in the International Standards Organization (ISO)-MPEG Audio Psychoacoustic Model 1 for Layer I [40]. The Layer I masking method is summarized as follows for a 32-kHz sampling rate. The MPEG model also supports sampling rates of 44.1 and 48 kHz.

The frequency mask is computed on localized segments (or windows) of the audio signal. The first step consists of computing the power spectrum of a short window (512 or 1024 samples) of the audio signal. Tonal (sinusoidal) and nontonal (noisy) components in the spectrum are identified because their masking models are different. A tonal component is a local maximum of the spectrum. The auditory system behaves as a bank of bandpass filters, with continuously overlapping center frequencies. These "auditory filters" can be approximated by rectangular filters with critical bandwidth increasing with frequency. In this model, the audible band is therefore divided into 24 nonregular critical bands.

Next, components below the absolute hearing threshold and tonal components separated by less than 0.5 Barks are removed. The final step consists of computing individual and global masking thresholds. The frequency axis is discretized according to hearing sensitivity and express frequencies in Barks. Note that hearing sensitivity is higher at low frequencies. The resulting masking curves are almost linear and depend on a masking index different for tonal and nontonal components. They are characterized by different lower and upper slopes depending on the distance between the masked and the masking component. We use f_1 to denote the set of frequencies present in the test signal. The global masking threshold for each frequency f_2 takes into account the absolute hearing threshold S_a and the masking curves P_2 of the N_t tonal components and N_n nontonal components

$$S_m(f_2) = 10 * \log_{10}\left[10^{S_a(f_2)/10} + \sum_{j=1}^{N_t} 10^{P_2(f_2,f_1,P_1)/10} + \sum_{j=1}^{N_n} 10^{P_2(f_2,f_1,P_1)/10}\right]. \tag{1}$$

AFFIDAVIT OF JAMES T. GRIFFIN

James T. ____ declares as follows:

1. I have been presented with a ZIP file entitled 1033.ZIP containing, among other files, another ZIP file entitled 1033-SRC.ZIP, which contains an executable program file entitled GPG.EXE and a document in Adobe Acrobat format entitled 1033-ART.PDF. After extracting GPG.EXE from 1033-SRC.ZIP, I executed it from the Windows 98/NT/2000 command line with the following text:

```
gpg --print-md sha1 1033.zip
```

I then saw a display appear in the program window with a string of alphanumeric characters. The characters were identical to those printed below this paragraph and repeated throughout the background of this entire paper as vertically oriented digits in various outline fonts.

8D06 7D30 7CDD 969E 3C3C C57D E9E3 467D 2E54 6AC3

2. I opened 1033-ART.PDF, which was also extracted from 1033.ZIP, and viewed it on my computer screen. I carefully compared the displayed document with an original artwork having images of flowers formed from torn-out pieces of paper, fixed to a piece of construction paper beneath a transparent laminating surface.

3. I personally confirmed that 1033-ART.PDF is a true and correct reproduction of the original artwork, to the degree of accuracy permitted by the displayed reproduction. The only visible discrepancies between the reproduction and the original artwork are (1) the coloration of the background, which appears as a washed-out green in the original artwork and has more of a purplish hue in the reproduction, and (2) the lower 1 inch or so of the original artwork is not shown in the reproduction.

I declare under penalty of perjury that the foregoing statements are true and correct.

Signed at ____________________, _______ This ______ day of ___________, ___________
City *State* *Date*

By: VOID
James T.______

This ________ day of ___________, _________, the above-named person personally came before me and executed the foregoing Affidavit in my presence.

Notary: Outline digits are in background of entire document. Please stamp over digits in space above.

Notary Public: VOID

G-1

APPENDIX H

FILING CONTENTS AUTHENTICATION

Joe Agent creates a document when filing a patent application for a client. He wants an additional piece of evidence that the patent application was mailed and that what was mailed was the patent application. So, he creates a PDF file of the application's text and drawings and computes the SHA1 hash of the PDF file using some the free GPG software ("GNU Privacy

Guard"), which he trusts. He makes a copy of the Express Mail label under which the application is to be mailed BEFORE having the label initialed and dated by the postal worker.

He then makes a Word 97 document that includes a brief block of text for a witness to sign and date. The block of text includes the Express Mail label and the hash. The document displays the digits of the hash as vertically oriented digits in various outline fonts, as

with the ACI (See, e.g., Appendix A.) He then puts the photocopy of the non-initialed express

mail label in his printer's paper supply, and prints the Word 97 (RTM) document.

The resulting document bears the image of the non-initialed express mail label with the hash of the patent application PDF file throughout its background and in a block of text of the top. Joe presents the document to Jane Attorney, an attorney in the office next door, for her dated signature. Now he can present the PDF file and the signed and dated paper document as evidence that the patent application reproduced in the PDF file existed before the express mail label was used to mail a package. This provides evidence that the patent application, reproduced in the PDF file, was actually what was mailed on the date of the express mail label. The proof can be made a bit

stronger by including the express mail label number in a footer on the first page of the patent application, so that the PDF file contains the express mail label number shown in the document. But that's going way beyond the level of evidence currently needed to reconstruct a file lost by the PTO.

This particular embodiment of the inventions can be employed any situation where proof of mailing of a particular document is desired, without the need for digital signatures and the associated hassles of public key authentication. The witness to this document doesn't even need to know what digital signatures or hash codes are. He or she is simply testifying as to the existence of the document with that particular code.

$$f(x,y,N)=\left(\frac{x-y}{x}\right)^N$$

$$f(x,y,N)=\exp\left(N\cdot\ln\left(\left|1-\frac{1}{x}\cdot y\right|\right)\right)\cdot\cos\left[N\cdot\left(\frac{1}{2}-\frac{1}{2}\cdot 1\right)\cdot\pi\right]\ \ldots$$
$$+\,i\cdot\exp\left(N\cdot\ln\left(\left|1-\frac{1}{x}\cdot y\right|\right)\right)\cdot\sin\left[N\cdot\left(\frac{1}{2}-\frac{1}{2}\cdot 1\right)\cdot\pi\right]$$

$$f(x,y,N)=\exp\left(N\cdot\ln\left(\left|1-\frac{1}{x}\cdot y\right|\right)\right)\cdot 1\ \ldots$$
$$+\,i\cdot\exp\left(N\cdot\ln\left(\left|1-\frac{1}{x}\cdot y\right|\right)\right)\cdot\sin\left[N\cdot\left(\frac{1}{2}-\frac{1}{2}\cdot 1\right)\cdot\pi\right]$$

$$f(x,y,N):=\exp\left(N\cdot\ln\left(\left|1-\frac{1}{x}\cdot y\right|\right)\right)$$

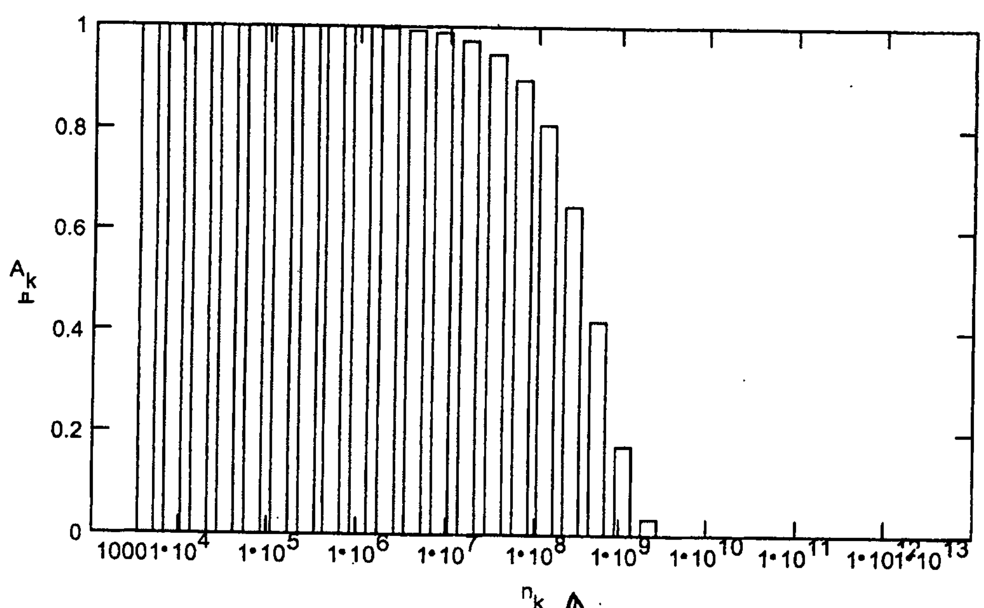
M

$x := 2^{32}$ $y := 7$ $\Leftarrow p = 2^{32} - 7$

$k := 12..40$ $n_k := 2^k$

$A_k := f(x, y, n_k)$

Graph: Product of probabilities of obtaining output changed, i.e., not a pass-through value, for each iteration, given n iterations.



50% chance of getting one pass-through value w/ random inputs after $10^8 - 10^9$ iterations.

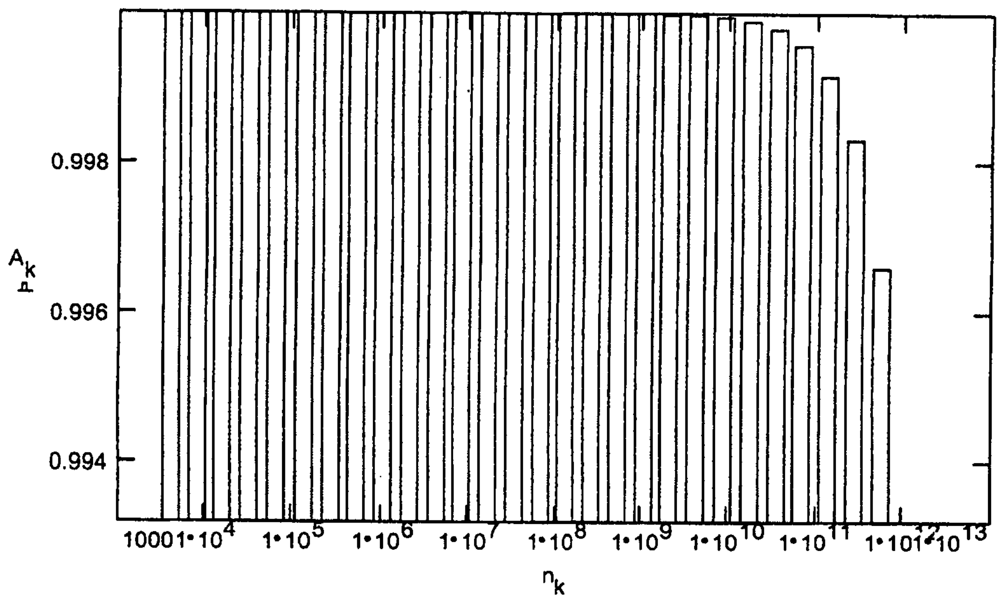
$x := 2^{50}$ ⟸ Larger M

$A_k := f(x, y, n_k)$ Same k

changed output, i.e., not q

Graph: Product of probabilities of obtaining pass-through value for each iteration, given n iterations

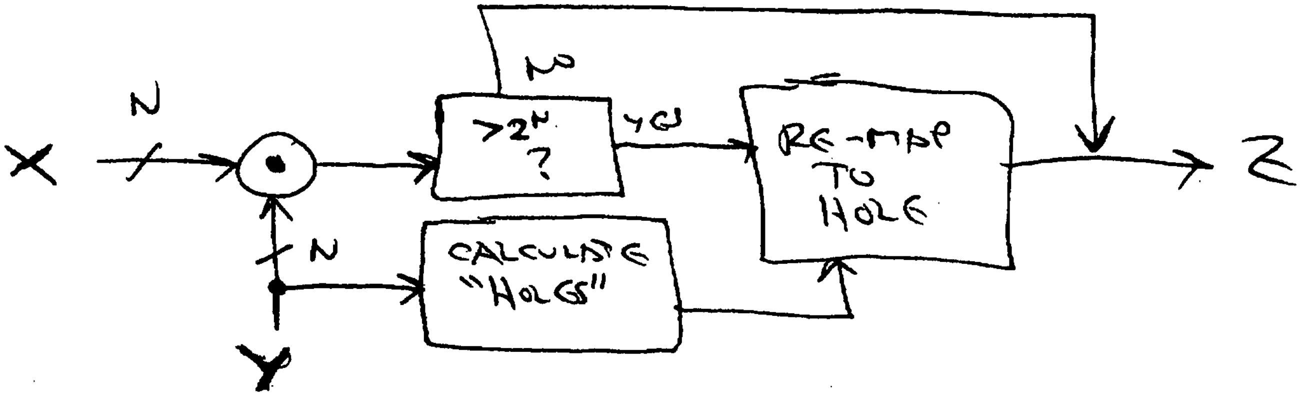
p = M − k



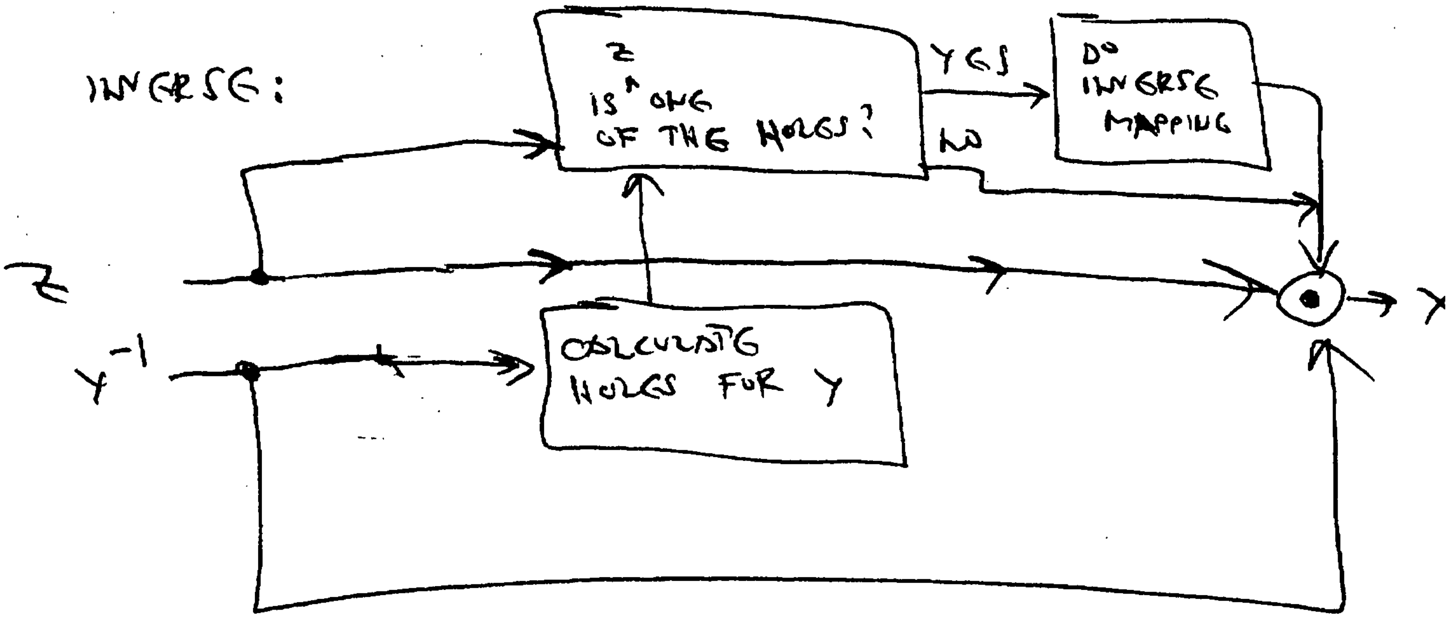
Need for exception handling can be made vanishingly unlikely.

With $p > 2^N$:

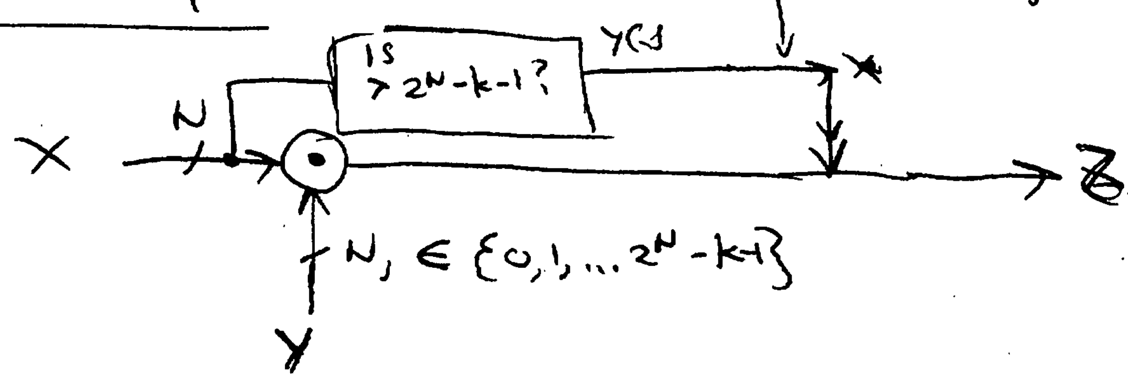
MODIFIED "⊙" TO WORK OUTSIDE $N = \{4, 8, 16\}$, $p = 2^N + 1$



INVERSE:



With $p < 2^N$



Interesting Primes (closest to 2^N)

$2^{20}+7$ $\quad$ $20^{20}-3$

$2^{32}+15$ $\quad$ $2^{32}-5$

2^N+1 not prime for $16 < N \leq 32$

$2^{16}+3$, -15

$2^{18}+3$, -5

$2^{19}-1$

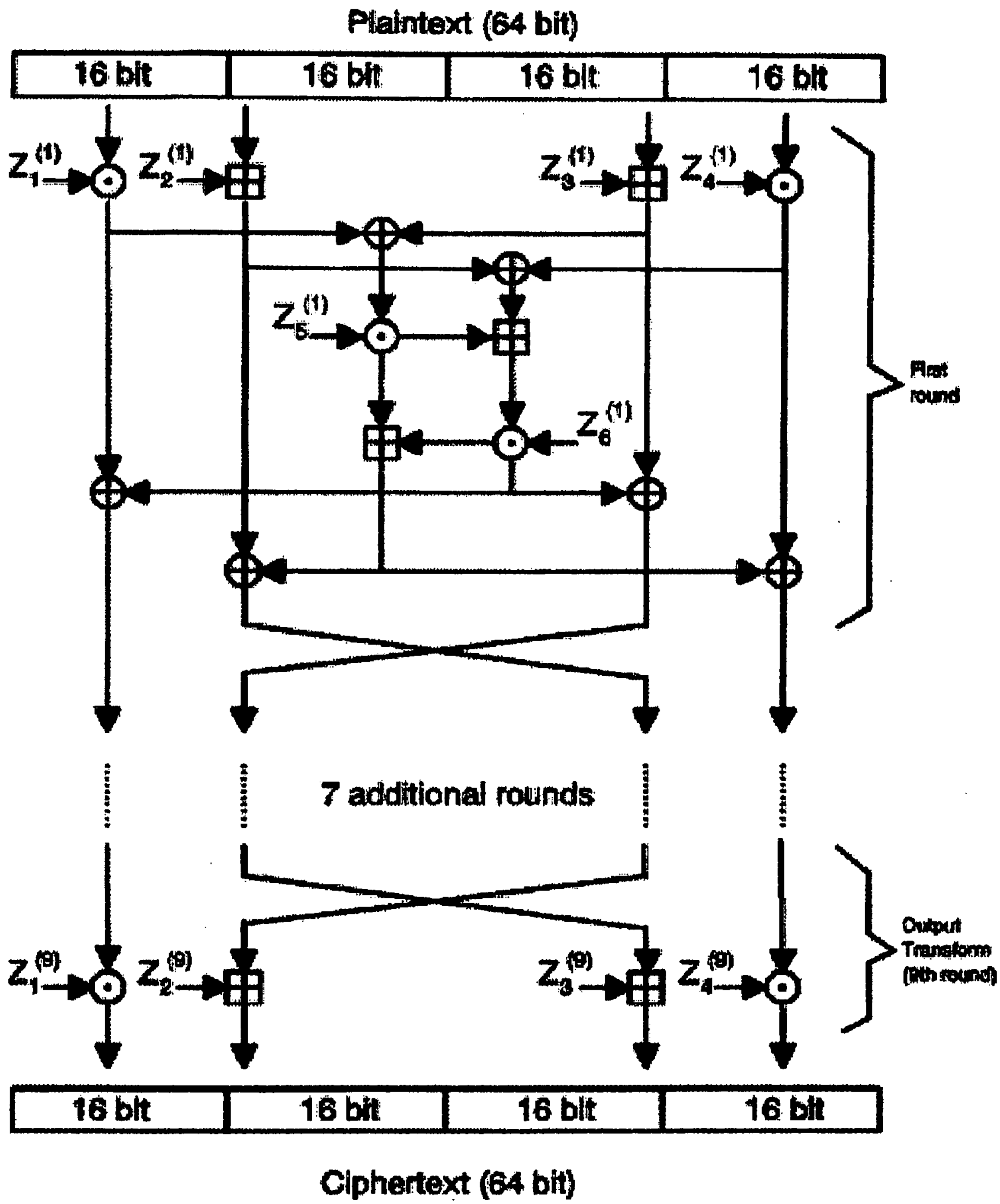
$2^{28}+3$

Bit Length	Test Number (Y=prime, N=not prime, blank=not tested)								
N	2^N-1	2^N-3	2^N-5	2^N-7	2^N-9	2^N-11	2^N-13	2^N-15	2^N-17
16	N	N	N	N	N	N	N	Y	
17	Y								
18	N	N							
19	Y								
20	N	Y							
21	N	N							
22	N	Y							
23	N	N							
24	N	Y							
25	N	N							
26	N	N							
27	N	N							
28	N	N							
29	N	Y							
30	N	N							
31	Y								
32	N	N	$\boxed{Y}$						
33	N	N	N	N	N				
34	N	N	N	N	N				
35	N	N	N	N	N				
36	N	N	Y						
37	N	N	N	N	N				
38	N	N	N	N	N				
39	N	N	N	Y					
40	N	N	N	N	N	N	N	N	N
41	N	N	N	N	N				
42	N	N	N	N	N				
43	N	N	N	N	N				
44	N	N	N	N	N				
45	N	N	N	N	N				
46	N	N	N	N	N				
47	N	N	N	N	N				
48	N	N	N	N	N	N	N	N	N

	Effective bit length for one "leaked" bit, per offset below 2^N								
2^N	1	3	5	7	9	11	13	15	17
65536								12.0	
131072	16.0								
262144									
524288	18.0								
1048576		18.0							
2097152									
4194304		20.0							
8388608									
16777216		22.0							
33554432									
67108864									
134217728									
268435456									
536870912		27.0							
1073741824									
2147483648	30.0								
4294967296			$\boxed{29.4}$						
8589934592									
17179869184									
34359738368									
68719476736			33.4						
1.37439E+11									
2.74878E+11									
5.49756E+11				36.0					
1.09951E+12									
2.19902E+12									
4.39805E+12									
8.79609E+12									
1.75922E+13									
3.51844E+13									
7.03687E+13									
1.40737E+14									
2.81475E+14									

Six ~~five~~ members of $\{0,1\}^{32}$ (4,294,967,290; 4...291; 4...292; 4...293; 4...294; 4...295) are passed through function w/o

Likelihood of pass-through is $6/2^{32}$ $= 1.4 \times 10^{-9}$



$\oplus$ Bit-by-bit exclusive OR of two 16-bit subblocks

$\boxplus$ Addition modulo2**16 of two 16 bit integers

$\odot$ Multiplication modulo2**16 + 1 of two 16-bit integers (subblock of all zeroes corresponds to 2**16)

IDEA CIPHER

From H.A.C. (Handbook of Applied Cryptography)

pseudo-group

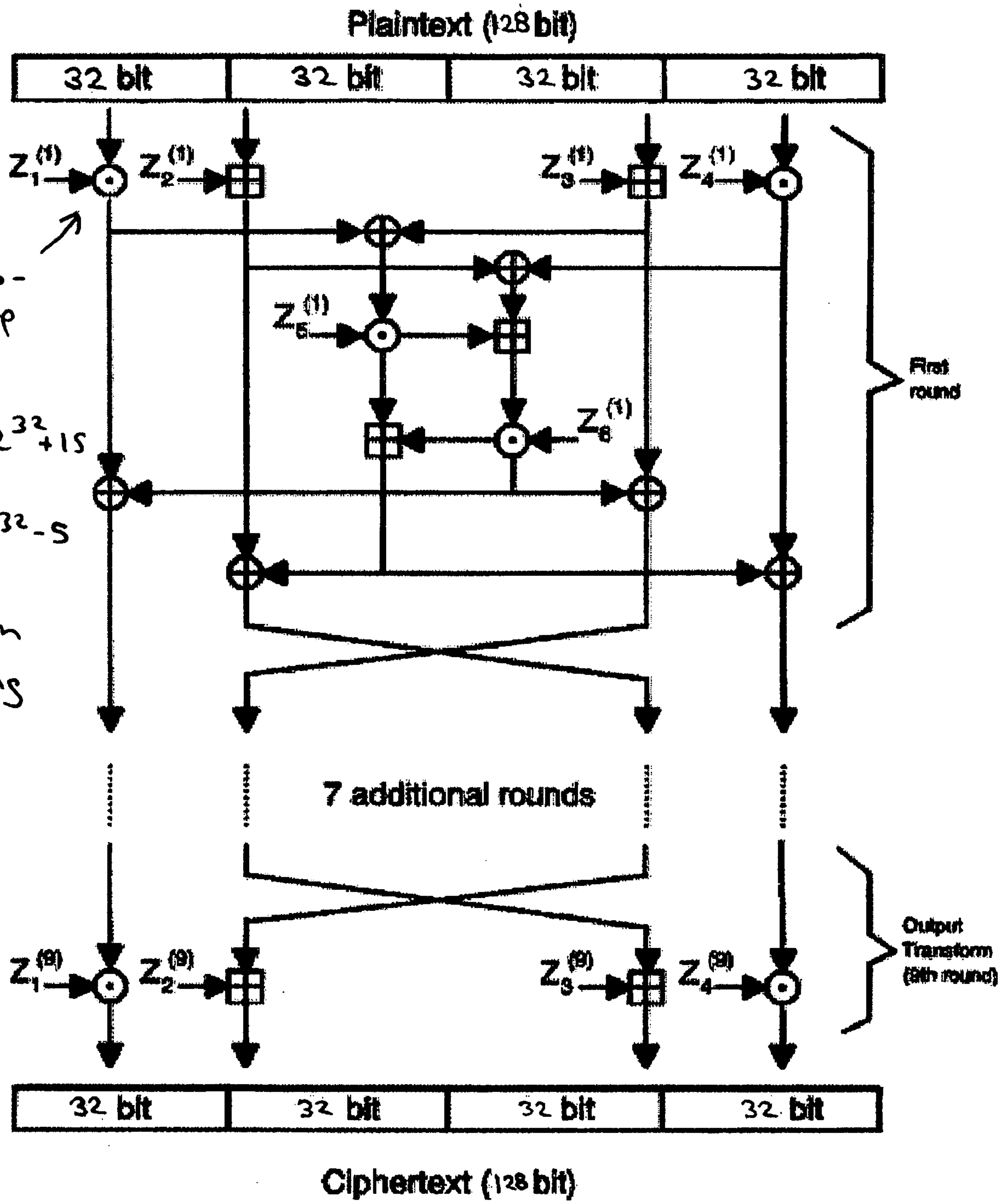
e.g.,

xy mod $2^{32}+15$

or

xy mod $2^{32}-5$

with exception handling



$\oplus$ Bit-by-bit exclusive OR of two 32-bit subblocks

$\boxplus$ Addition modulo2**32 of two 32 bit integers

$\odot$ Multiplication modulo2** 32 - 5 of two 32-bit integers

Adapted from H.A.C.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	*13*	*14*	*15*	*16*
2	2	4	6	8	10	12	1	3	5	7	9	11	*13*	*14*	*15*	*16*
3	3	6	9	12	2	5	8	11	1	4	7	10	*13*	*14*	*15*	*16*
4	4	8	12	3	7	11	2	6	10	1	5	9	*13*	*14*	*15*	*16*
5	5	10	2	7	12	4	9	1	6	11	3	8	*13*	*14*	*15*	*16*
6	6	12	5	11	4	10	3	9	2	8	1	7	*13*	*14*	*15*	*16*
7	7	1	8	2	9	3	10	4	11	5	12	6	*13*	*14*	*15*	*16*
8	8	3	11	6	1	9	4	12	7	2	10	5	*13*	*14*	*15*	*16*
9	9	5	1	10	6	2	11	7	3	12	8	4	*13*	*14*	*15*	*16*
10	10	7	4	1	11	8	5	2	12	9	6	3	*13*	*14*	*15*	*16*
11	11	9	7	5	3	1	12	10	8	6	4	2	*13*	*14*	*15*	*16*
12	12	11	10	9	8	7	6	5	4	3	2	1	*13*	*14*	*15*	*16*
13	ILLEGAL KEY VALUES															
14																
15																
16																

$N = 4, \ \rho = 2^N - 3$

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	*7*	*8*
2	2	4	6	1	3	5	*7*	*8*
3	3	6	2	5	1	4	*7*	*8*
4	4	1	5	2	6	3	*7*	*8*
5	5	3	1	6	4	2	*7*	*8*
6	6	5	4	3	2	1	*7*	*8*
7	ILLEGAL KEY VALUES							
8								

$N = 3, \ \rho = 2^N - 1$

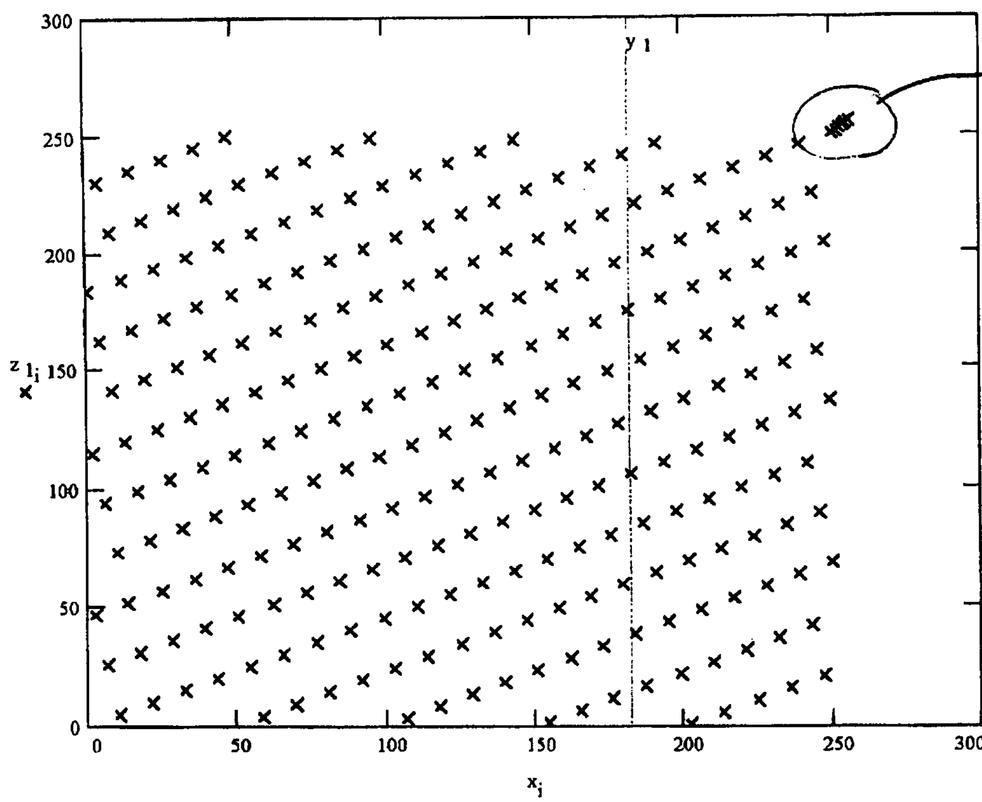
	1	2	3	. . .	20,537	20,538	20,539	. . .	16,777,211	16,777,212	16,777,213	16,777,214	16,777,215	16,777,216
1	1	2	3	. . .	20,537	20,538	20,539	. . .	16,777,211	16,777,212	16,777,213	16,777,214	16,777,215	16,777,216
2	2	4	6	. . .	41,074	41,076	41,078	. . .	16,777,209	16,777,211	16,777,213	16,777,214	16,777,215	16,777,216
3	3	6	9	. . .	61,611	61,614	61,617	. . .	16,777,207	16,777,210	16,777,213	16,777,214	16,777,215	16,777,216
⋮	⋮	⋮	⋮		⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	⋮
20,537	20,537	41,074	61,611	. . .	2,338,044	2,358,581	2,379,118	. . .	16,736,139	16,756,676	16,777,213	16,777,214	16,777,215	16,777,216
20,538	20,538	41,076	61,614	. . .	2,358,581	2,379,119	2,399,657	. . .	16,736,137	16,756,675	16,777,213	16,777,214	16,777,215	16,777,216
20,539	20,539	41,078	61,617	. . .	2,379,118	2,399,657	2,420,196	. . .	16,736,135	16,756,674	16,777,213	16,777,214	16,777,215	16,777,216
⋮	⋮	⋮	⋮		⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	⋮
16,777,211	16,777,211	16,777,209	16,777,207	. . .	16,736,139	16,736,137	16,736,135	. . .	4	2	16,777,213	16,777,214	16,777,215	16,777,216
16,777,212	16,777,212	16,777,211	16,777,210	. . .	16,756,676	16,756,675	16,756,674	. . .	2	1	16,777,213	16,777,214	16,777,215	16,777,216
16,777,213	ILLEGAL KEY VALUES													
16,777,214														
16,777,215														
16,777,216														

$N = 24, \ \varphi = 2^N - 3$

Multiplicative Group O, ation (pseudogroup)
f(x,y) = xy mod 2^N-k, where x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

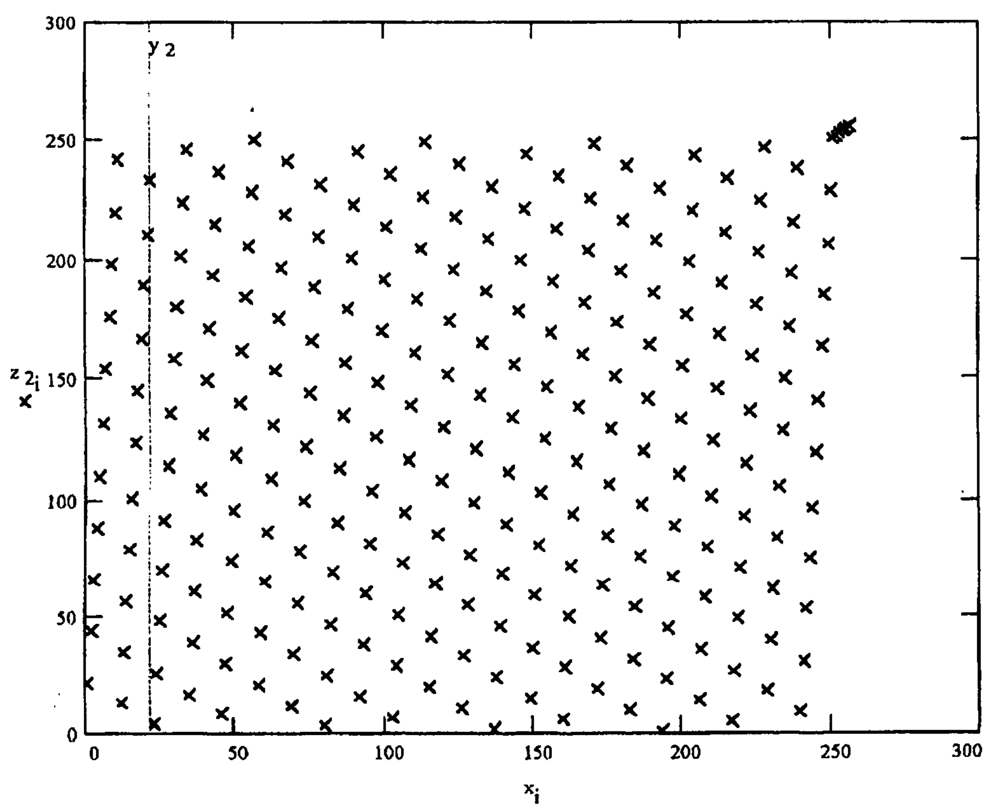
$N = 8$ bits $2^N = 256$ $k = 5$ $p = 251$ prime modulus

Pseudogroup operation with key y1: $y_1 = 183$ $z_{1_i} := f(x_i, y_1)$



Linear exception region per ESS 'invention', where $x > p-1$

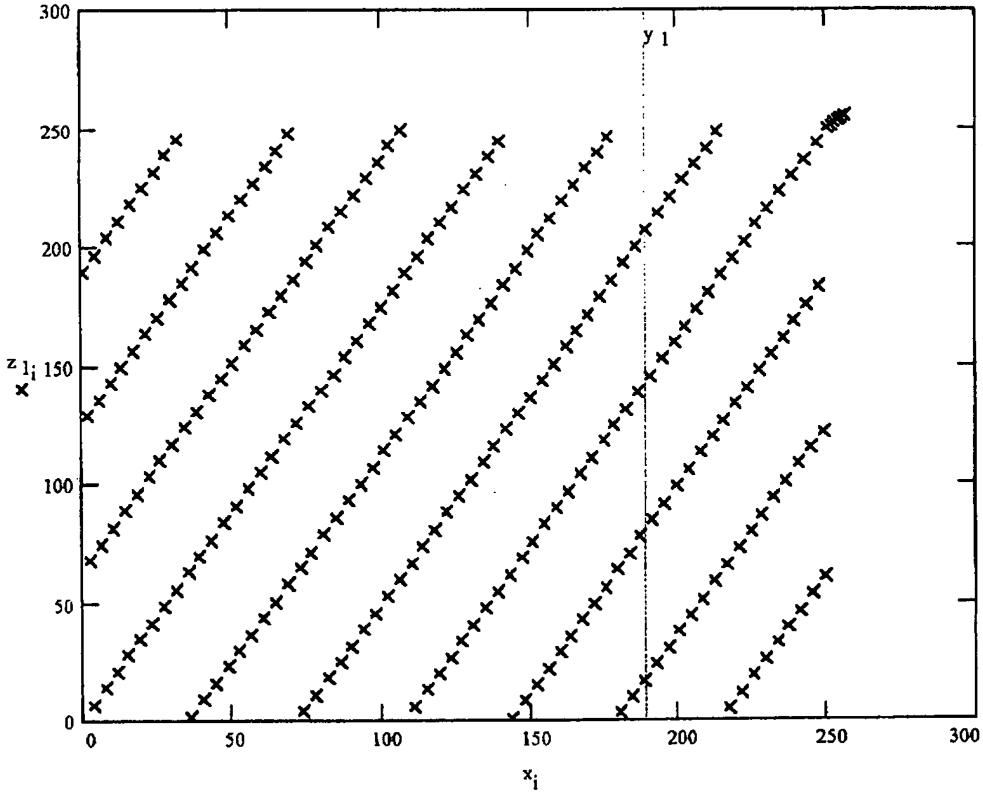
psedogroup operation with key y2: $y_2 = 22$ $z_{2_i} := f(x_i, y_2)$



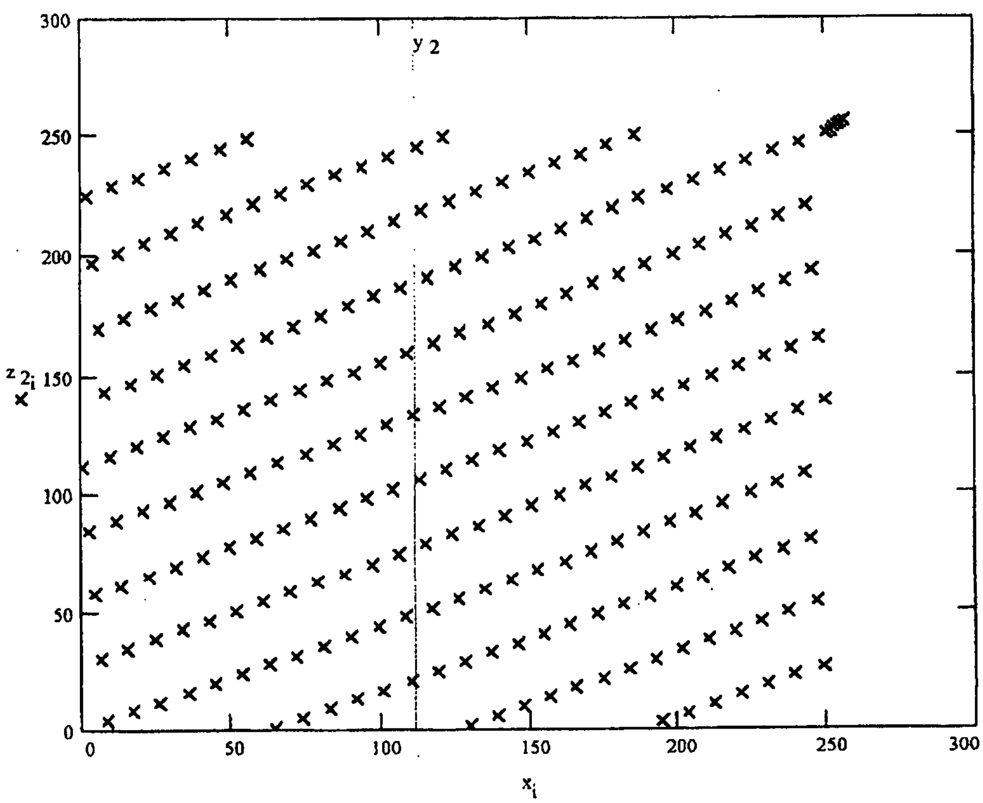
Multiplicative Group O   .ion (pseudogroup)
f(x,y) = xy mod 2^N-k, wnere x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

$N = 8$ bits $2^N = 256$ $k = 5$ $p = 251$ prime modulus

pseudogroup operation with key y1: $y_1 = 190$ $z_{1_i} := f(x_i, y_1)$



pseudogroup operation with key y2: $y_2 = 112$ $z_{2_i} := f(x_i, y_2)$

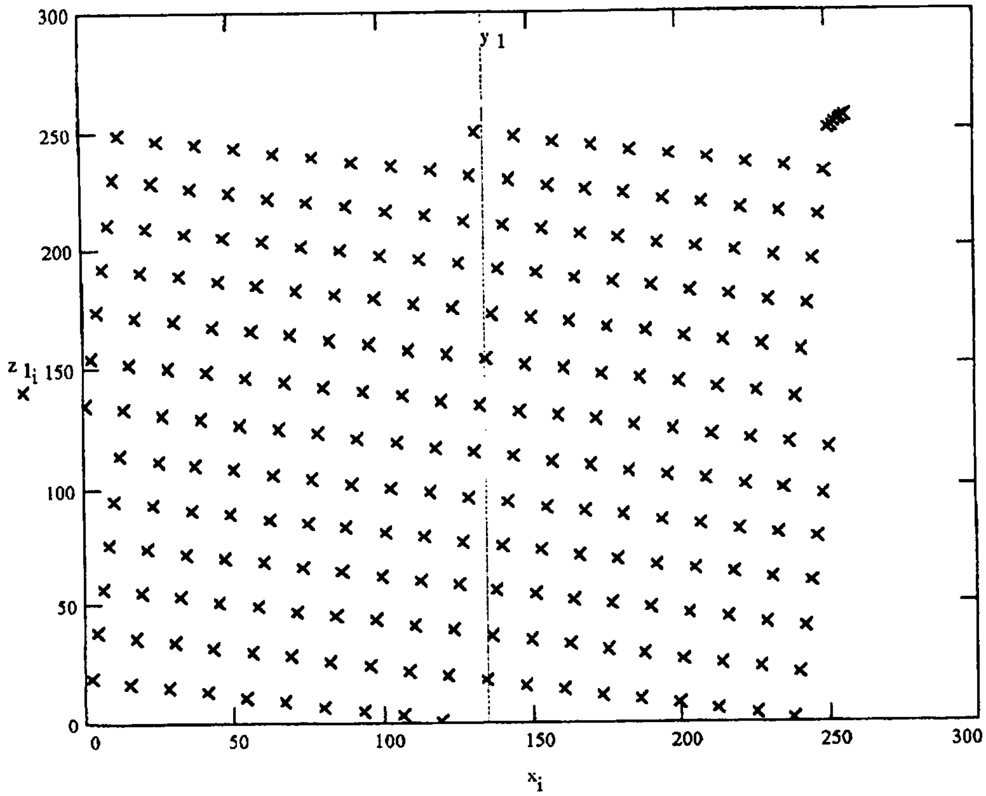


J-9

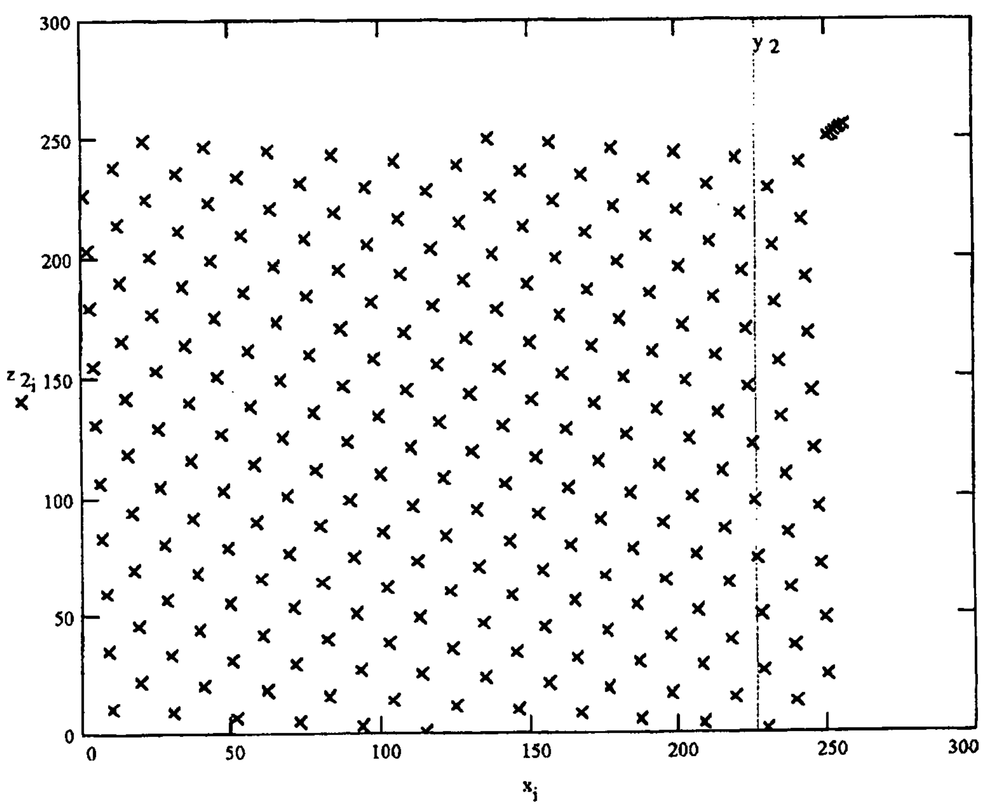
Multiplicative Group O ...ion (pseudogroup)
f(x,y) = xy mod 2^N-k, wnere x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

$N = 8$ bits $2^N = 256$ $k = 5$ $p = 251$ prime modulus

pseudogroup operation with key y1: $y_1 = 135$ $z_{1_i} := f(x_i, y_1)$



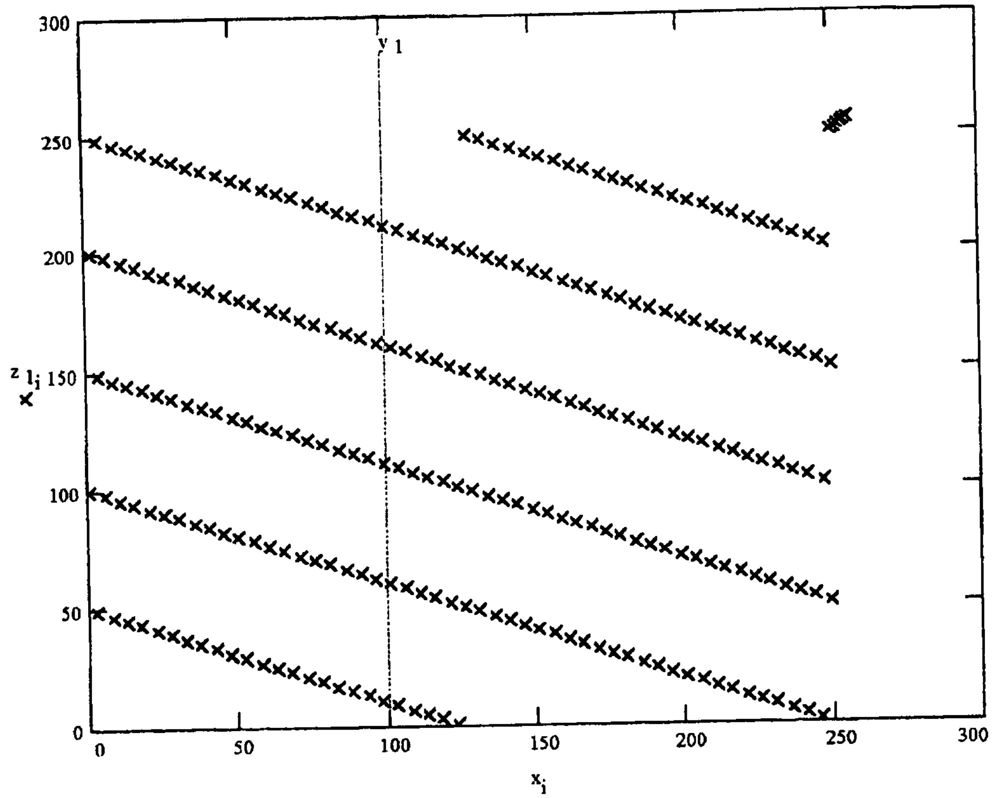
pseudogroup operation with key y2: $y_2 = 227$ $z_{2_i} := f(x_i, y_2)$



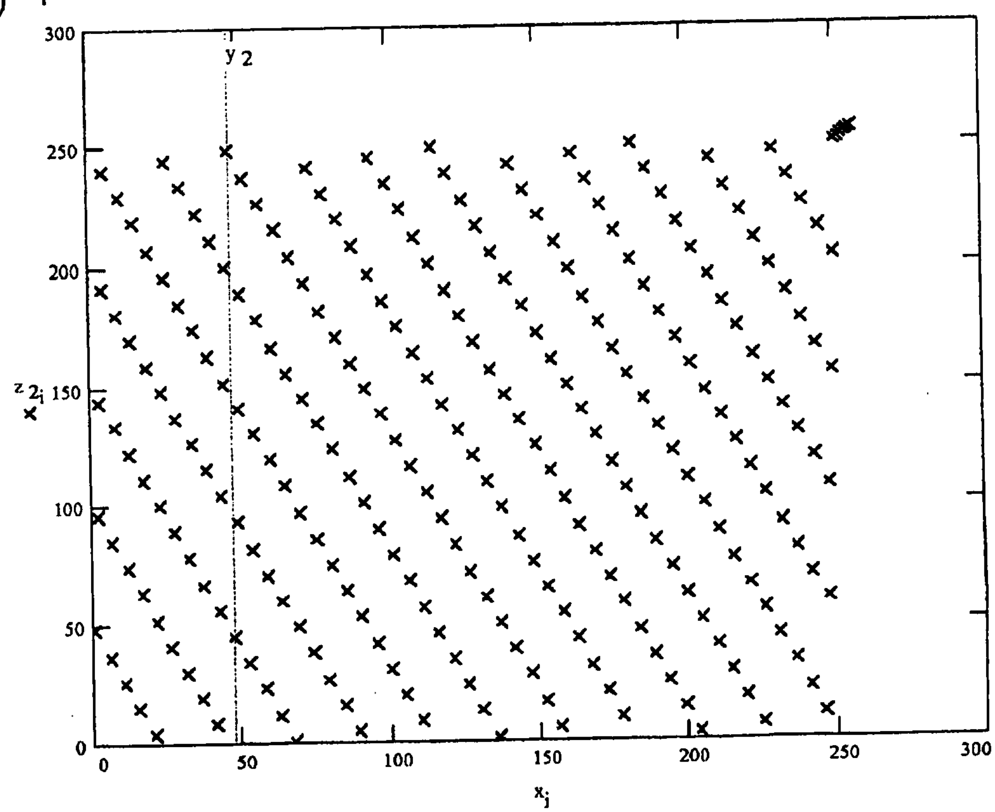
Multiplicative Group O   .tion (pseudogroup)
f(x,y) = xy mod 2^N-k, where x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

$N = 8$ bits $2^N = 256$ $k = 5$ $p = 251$ prime modulus

Pseudogroup operation with key y1: $y_1 = 100$ $z_{1_i} := f(x_i, y_1)$



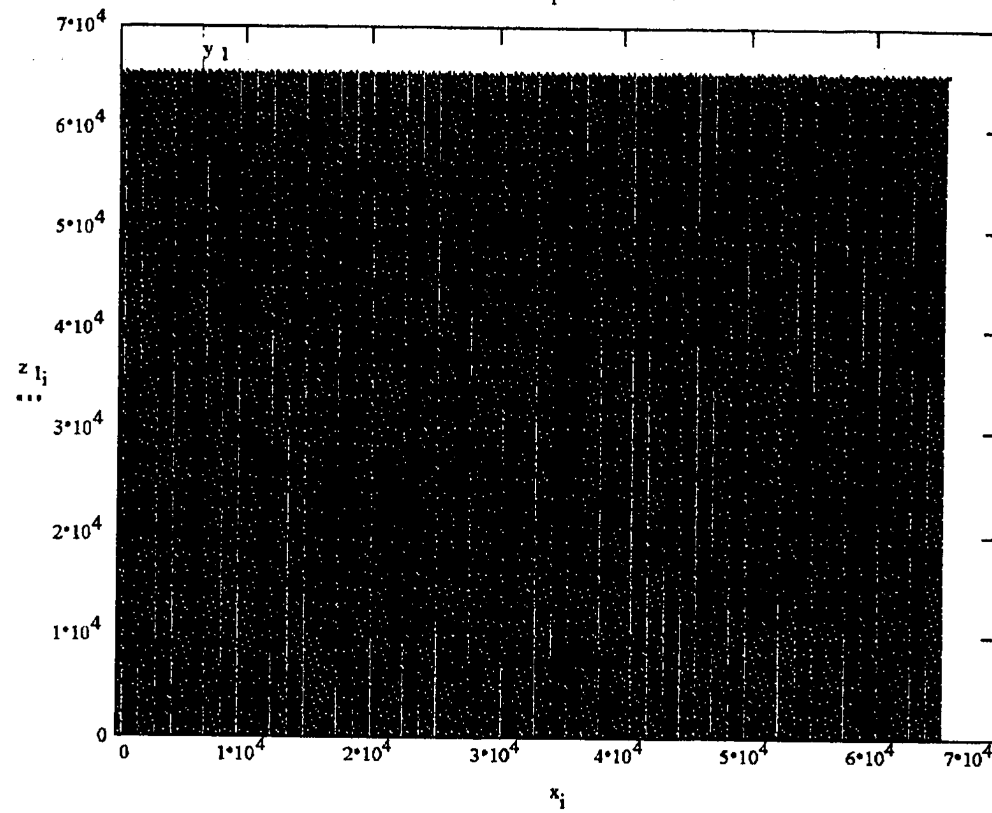
Pseudogroup operation with key y2: $y_2 = 48$ $z_{2_i} := f(x_i, y_2)$



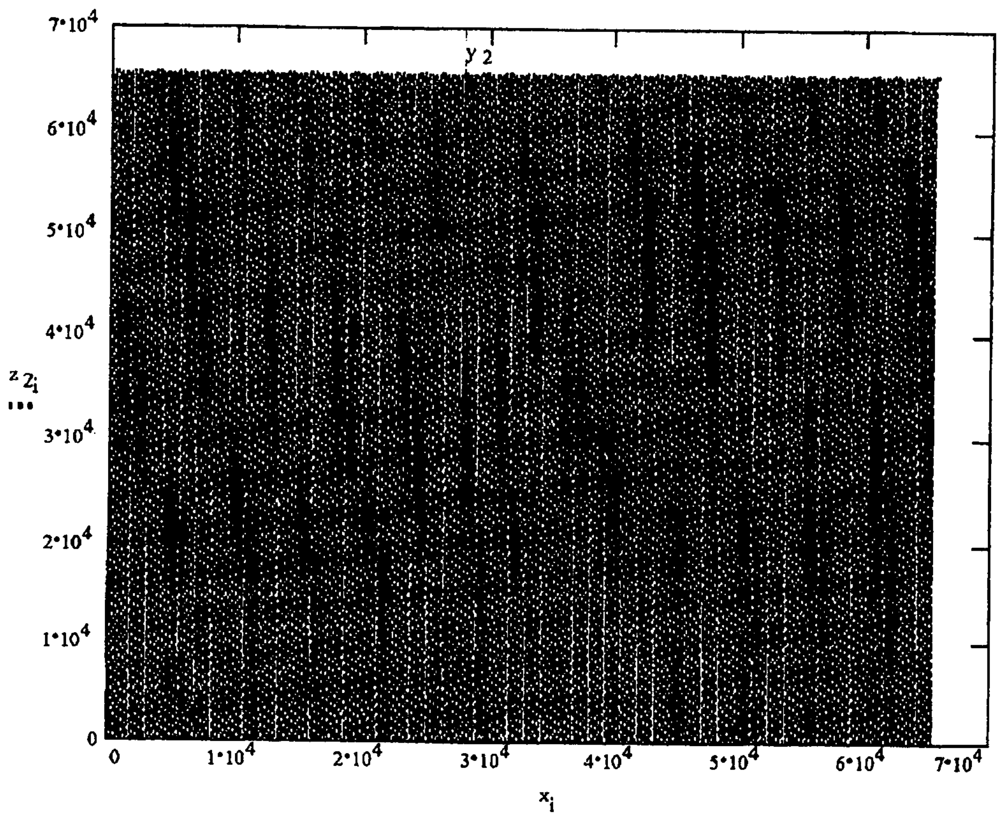
Multiplicative Group O .ion (pseudogroup)
f(x,y) = xy mod 2^N-k, wnere x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

N = 16 bits $2^N = 65536$ $k = 15$ $p = 65521$ prime modulus

pseudogroup operation with key y1: $y_1 = 6595$ $z_{1_i} := f(x_i, y_1)$

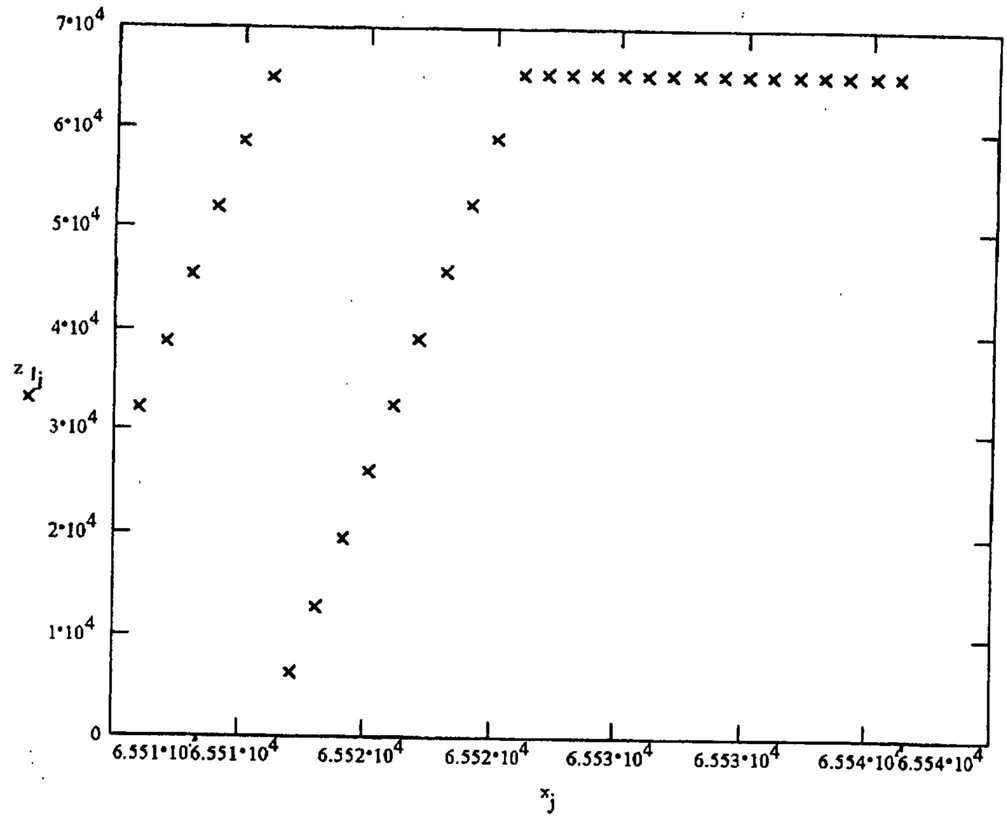


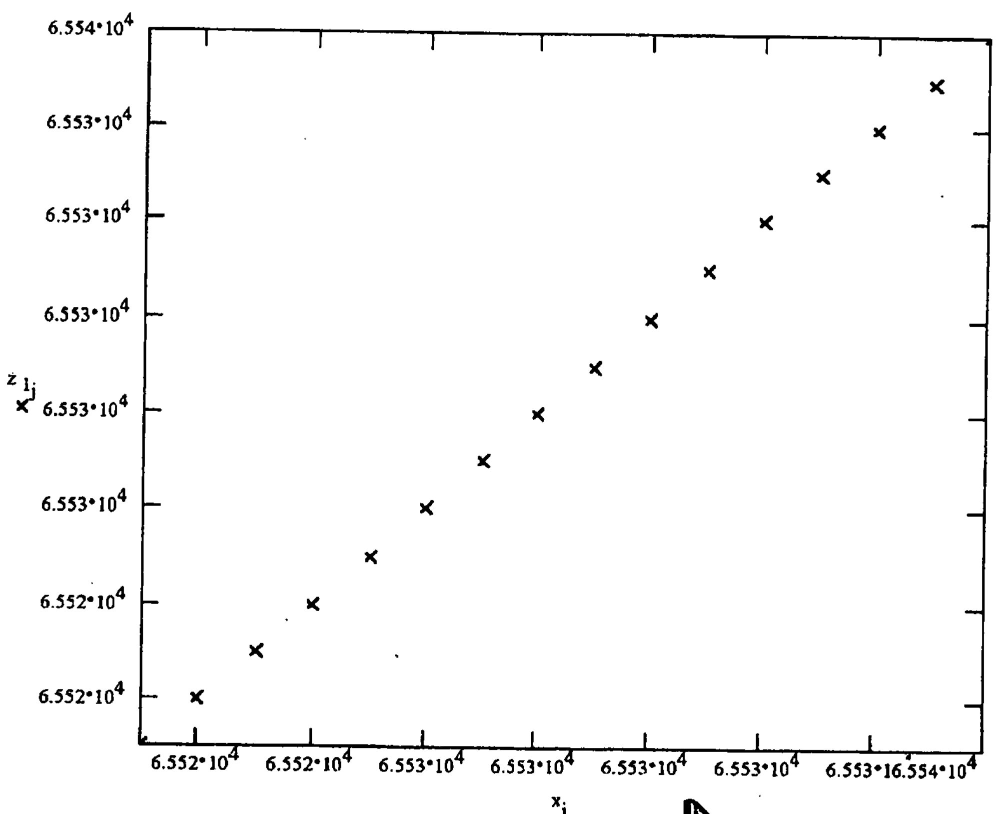
pseudogroup operation with key y2: $y_2 = 27939$ $z_{2_i} := f(x_i, y_2)$



Zoom in on exception region of plot with key Y1

$j := 2^N - 2 \cdot k .. 2^N$



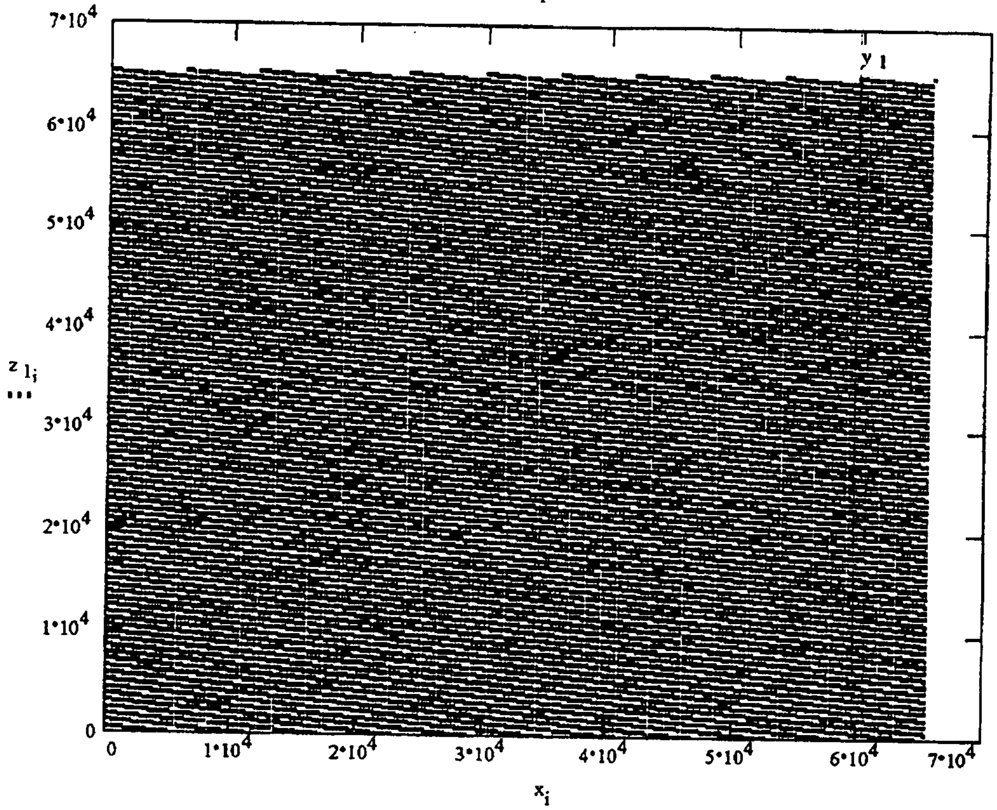


Linear Exception Region

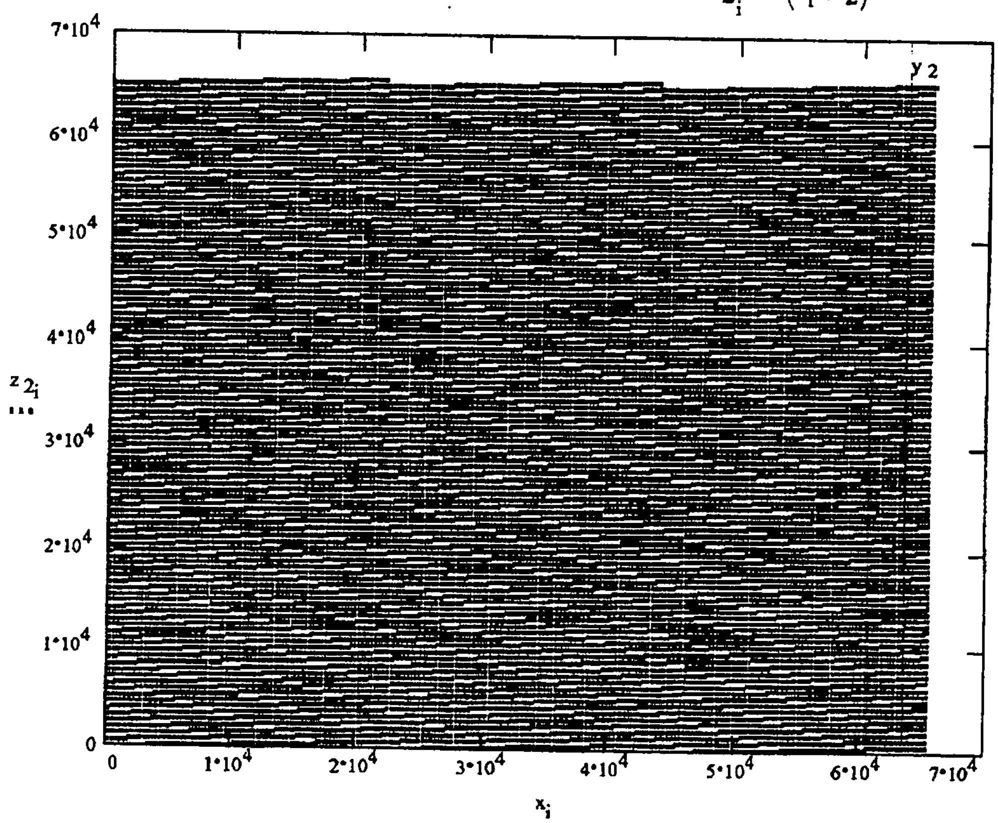
Multiplicative Group O ...tion (pseudogroup)
f(x,y) = xy mod 2^N-k, where x,y ^N-k;
= x where x < 2^N-k. (y always < 2^N-k.)

$N = 16$ bits $\qquad 2^N = 65536 \qquad k = 15 \qquad p = 65521$ prime modulus

pseudogroup operation with key y1: $\quad y_1 = 59624 \quad z_{1_i} := f(x_i, y_1)$

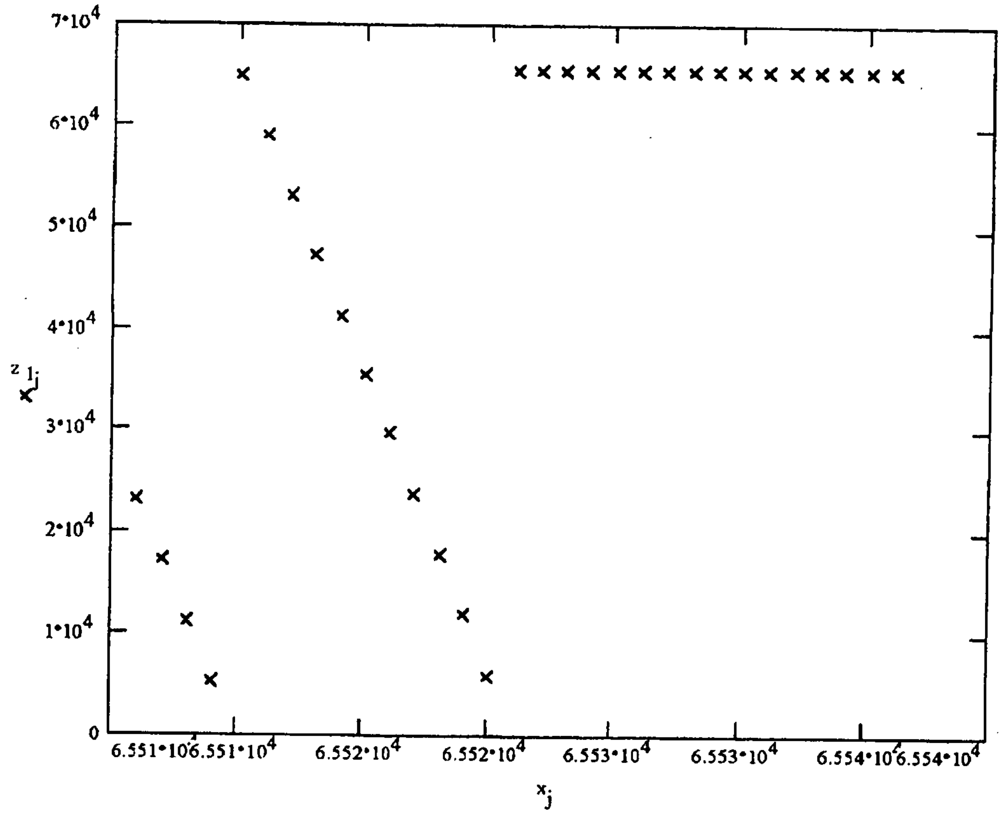


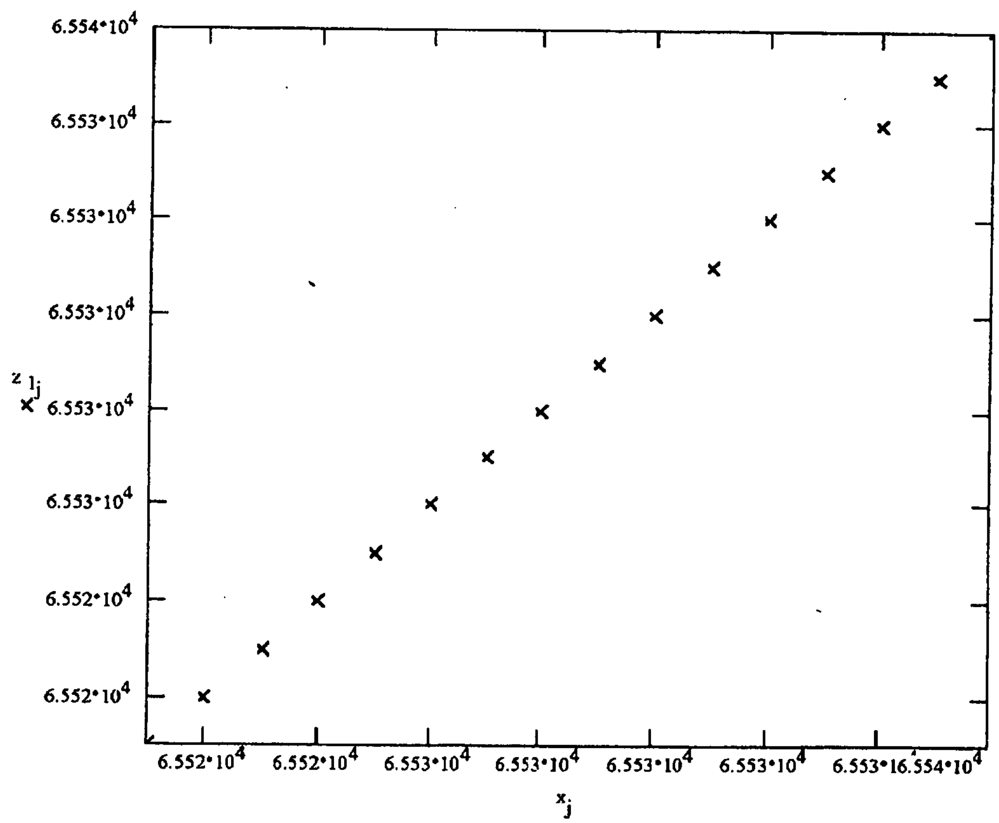
pseudogroup operation with key y2: $\quad y_2 = 63631 \quad z_{2_i} := f(x_i, y_2)$



Zoom in on exception region of plot with key Y1

$j := 2^N - 2 \cdot k .. 2^N$





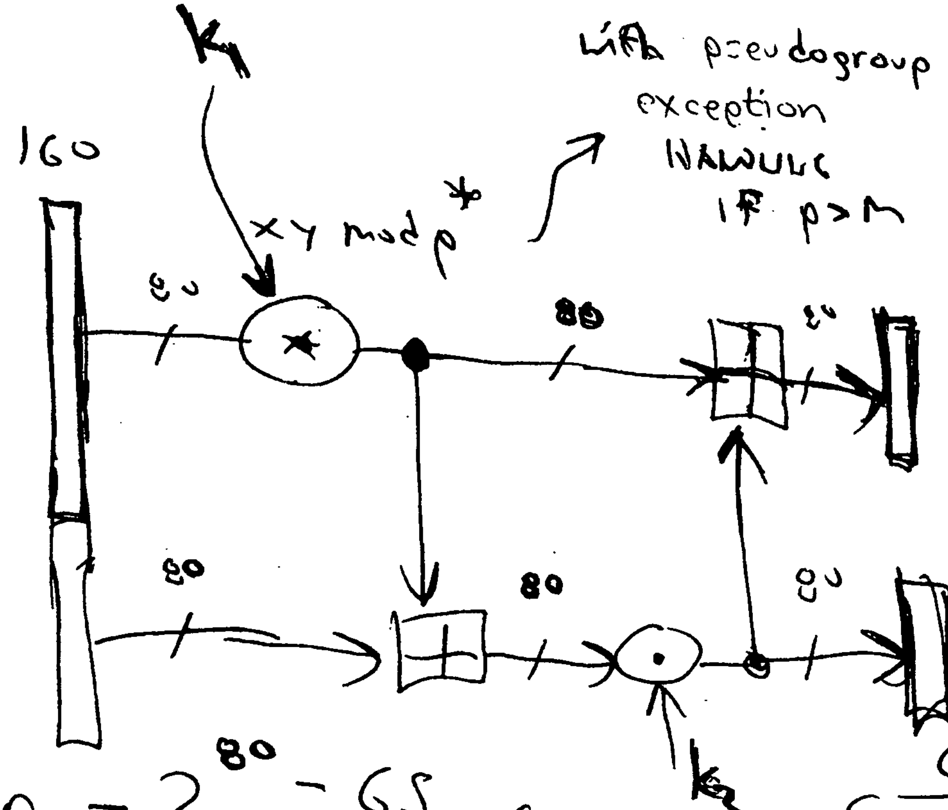
A post to sci.crypt suggested that increasing the IDEA subblock to 32-bit subblocks from the design of 16 bits would increase the secu of the IDEA algorithm to a factor of 2^{32}. Lai answered that the stren of the algorithm was based on the fact that $2^{16} + 1$ is a prime, whe $2^{32}+1$ is not. Lai suggests that the stronger properties of the algori would be compromised. The point is that small changes in structure have adverse ripple effects on the cryptographic structure that become serious implementation errors. We look at other implementa errors in Chapter 13.

Not so limited with pseudogroup

$xy \bmod 2^N \pm k$

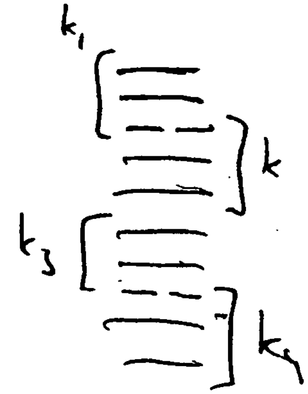
modification to ⊙

J-16



with pseudogroup exception HANDLING if $p > n$

JUST set MSB = 0 if invalid



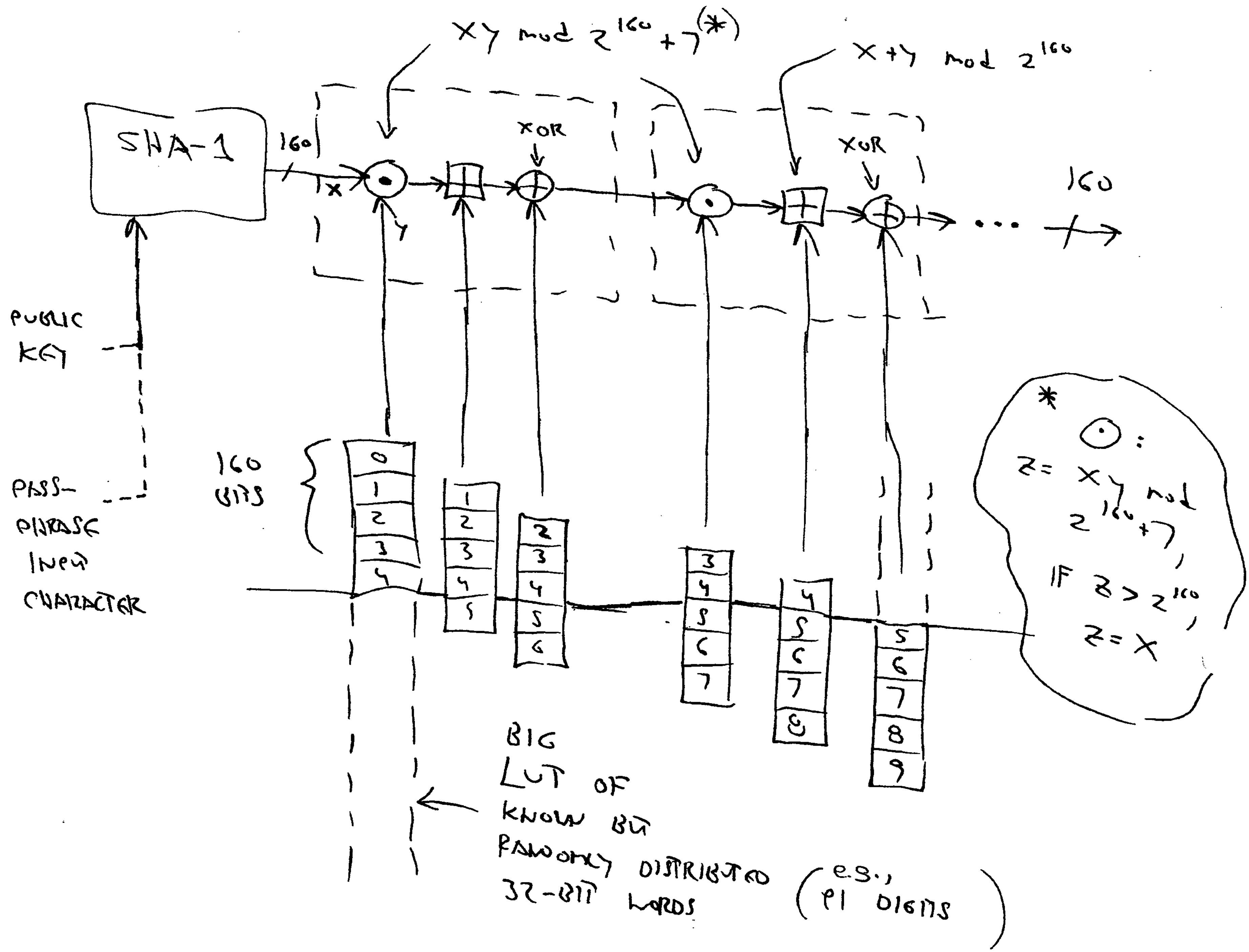
$$\frac{65}{2^{80}} = 1 \text{ of every } 1.9 \times 10^{22} \text{ keys is invalid}$$

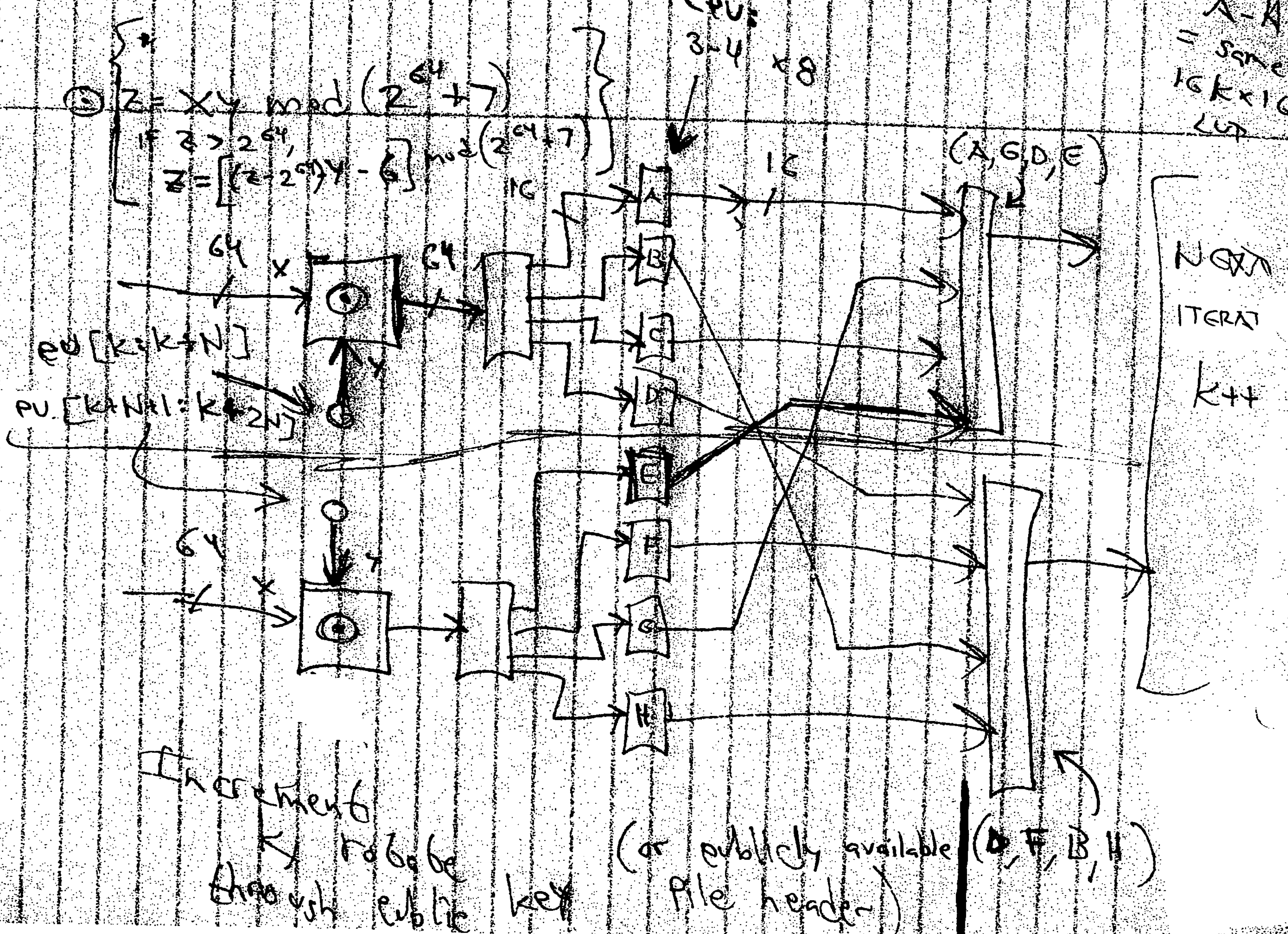
$p = 2^{80} - 65$ (prime)

* $p = 2^{80} + 13$ (prime)

will need to rarely compute holes for key

$(p-64 < \text{key} \leq p)$


SHA-1
160
X
Y
XY mod $2^{160}+7$ (*)
X+Y mod 2^{160}
XOR
XOR
160
PUBLIC KEY
PASS-PHRASE (NEW CHARACTER
160 BITS
BIG LUT OF KNOWN BUT RANDOMLY DISTRIBUTED 32-BIT WORDS (e.g., PI DIGITS)
* ⊙ : Z = XY mod $2^{160}+7$, IF Z > 2^{160}, Z = X

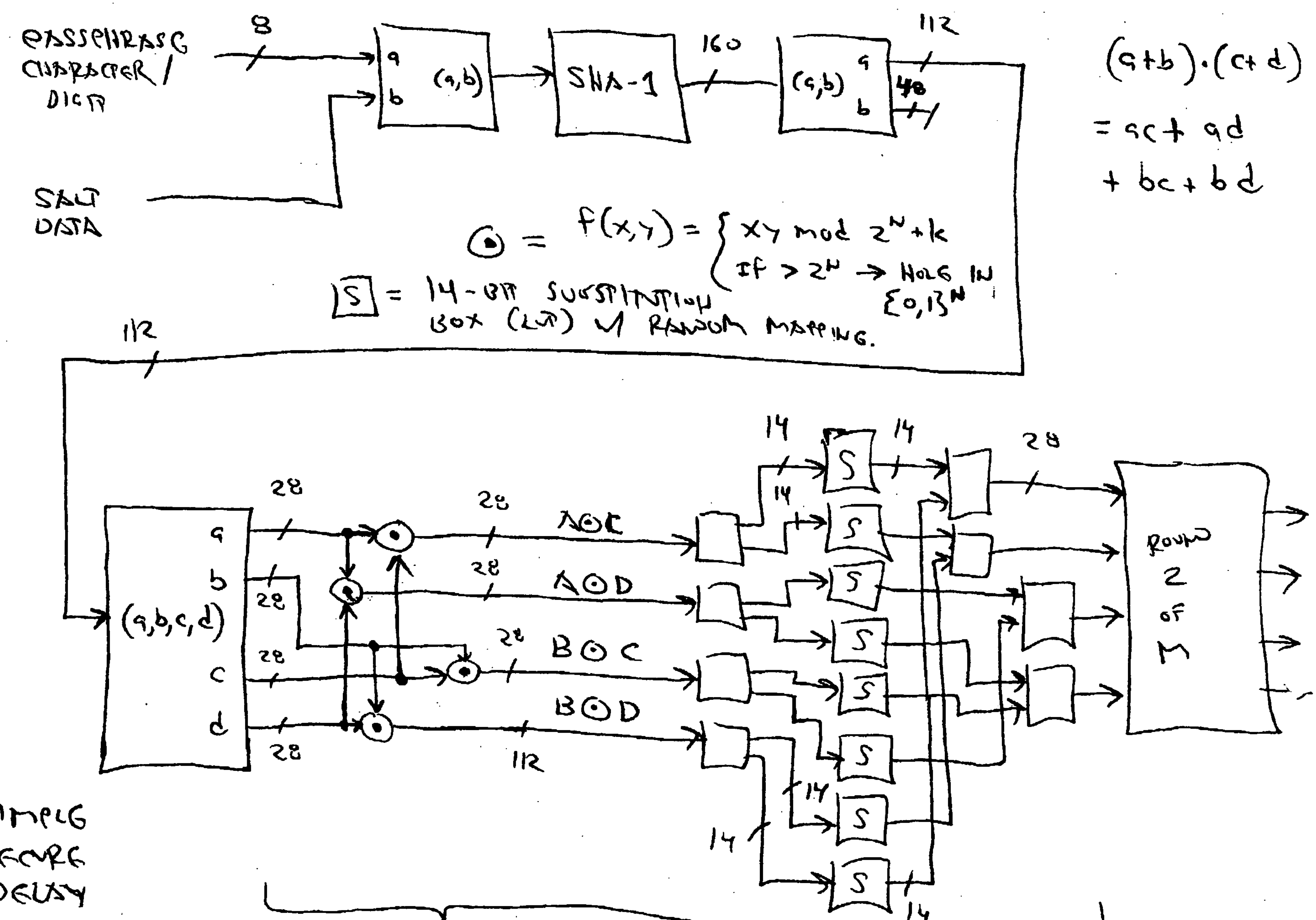


$\odot$ $Z = XY \bmod (2^{64}+7)$
if $Z > 2^{64}$,
$Z = [(Z-2^{64})Y - 6] \bmod (2^{64}+7)$
CPU: 3-4 x8
A-H = same 16k x16 LUT
64
X
Y
16
eU[k:k+N]
eU[k+N+1:k+2N]
A
B
C
D
E
F
G
H
(A, G, D, E)
(D, F, B, H)
NEXT ITERAT
k++
Increment k, to be through public key (or publicly available file header)

$(a+b)\cdot(c+d)$

$= ac + ad$

$+ bc + bd$



$\odot = f(x,y) = \begin{cases} xy \bmod 2^N + k \\ \text{if} > 2^N \rightarrow \text{HOLE IN } \{0,1\}^N \end{cases}$

$\boxed{S}$ = 14-BIT SUBSTITUTION BOX (LUT) W/ RANDOM MAPPING.

ROUND 1 OF M

16K × 16, ALL $\boxed{S}$ SAME (?)

SIMPLE SECURE DELAY

L-1

N (Bit Length)	2^N+1	2^N+3	2^N+5	2^N+7	2^N+9	2^N+11	2^N+13	2^N+15	2^N+51
	Test Number (Y=prime, N=not prime, blank=not tested)								
16	Y	Y							
17	N								
18	N	Y							
19	N								
20	N	N	N	Y					
21	N								
22	N								
23	N								
24	N	N	N	N	N	N	N	N	
25	N								
26	N								
27	N								
28	N	Y							
29	N								
30	N								
31	N								
32	N	N	N	N	N	N	N	Y	
40	N								
48	N	N	N	N	N	N	N	N	
56	N	N	N	N	N	N	N	N	
64	N	N	N	N	N	N	Y		
96	N								
128	N	N	N	N	N	N	N	N	(+51)

2^N	1	3	5	7	9	11	13	15	51
	Likelihood of Exception (-bits)								
65536	15.0	14.0							
131072									
262144		16.0							
524288									
1048576				17.0					
2097152									
4194304									
8388608									
16777216									
33554432									
67108864									
134217728									
268435456		26.0							
536870912									
1073741824									
2147483648									
4294967296								28.0	
1.09951E+12									
7.20576E+16									
1.84467E+19									
7.92282E+28							60.2		
3.40282E+38									
#REF!									122.3
1									
1									
1									
1									
1									
1									
1									
1									
1									
1									

M-1

Bit Length	Test Number (Y=prime, N=not prime, blank=not tested)								
N	2^N-1	2^N-3	2^N-5	2^N-7	2^N-9	2^N-11	2^N-13	2^N-15	2^N-17
16	N	N	N	N	N	N	N	Y	
17	Y								
18	N	N							
19	Y								
20	N	Y							
21	N	N							
22	N	Y							
23	N	N							
24	N	Y							
25	N	N							
26	N	N							
27	N	N							
28	N	N							
29	N	**Y**							
30	N	N							
31	**Y**								
32	N	N	**Y**						
33	N	N	N	N	N				
34	N	N	N	N	N				
35	N	N	N	N	N				
36	N	N	**Y**						
37	N	N	N	N	N				
38	N	N	N	N	N				
39	N	N	N	Y					
40	N	N	N	N	N	N	N	N	N
41	N	N	N	N	N				
42	N	N	N	N	N				
43	N	N	N	N	N				
44	N	N	N	N	N				
45	N	N	N	N	N				
46	N	N	N	N	N				
47	N	N	N	N	N				
48	N	N	N	N	N	N	N	N	N

	Effective bit length for one "leaked" bit, per offset below 2^N								
2^N	1	3	5	7	9	11	13	15	17
65536								12.0	
131072	16.0								
262144									
524288	18.0								
1048576		18.0							
2097152									
4194304		20.0							
8388608									
16777216		22.0							
33554432									
67108864									
134217728									
268435456									
536870912		**27.0**							
1073741824									
2147483648	**30.0**								
4294967296			**29.4**						
8589934592									
17179869184									
34359738368									
68719476736			**33.4**						
1.37439E+11									
2.74878E+11									
5.49756E+11				36.0					
1.09951E+12									
2.19902E+12									
4.39805E+12									
8.79609E+12									
1.75922E+13									
3.51844E+13									
7.03687E+13									
1.40737E+14									
2.81475E+14									

TABLE I-1: "Pseudogroup" Operation: p=11 (prime), m=3 bits

— KEY VALUES →

INPUT VALUES ↓

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	10	1	3	5
3	3	6	9	1	4	7	10	2
4	4	8	1	5	9	2	6	10
5	5	10	4	9	3	8	2	7
6	6	1	7	2	8	3	9	4
7	7	3	10	6	2	9	5	1
8	8	5	2	10	7	4	1	9

TABLE I-2: Key values AND holes associated with them

Each row contains input values producing a given output, except for holes (black squares).
Each column contains key values produc ing outputs for a given input, except for holes.

— KEY VALUES →

Output VALUES ↓

	1	2	3	4	5	6	7	8
1	1	6	4	3	■	2	8	7
2	2	1	8	6	7	4	5	3
3	3	7	1	■	5	6	2	■
4	4	2	5	1	3	8	■	6
5	5	8	■	4	1	■	7	2
6	6	3	2	7	■	1	4	■
7	7	■	6	■	8	3	1	5
8	8	4	■	2	6	5	■	1

A "hole" is an output value that will not occur for any in the set {1,2,...2^m} of possible key values, given a particular input value in that set.

Note from table I-1 and I-2 that the output value of 7 does not occur with Key value of 2.

TABLE II-1:(PRIOR ART) **Multiplicative Group Operation x*y mod p, p=17 (prime), m=4 bits**

— KEY VALUES →

INPUT VALUES ↓

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	*16*
2	2	4	6	8	10	12	14	16	1	3	5	7	9	11	13	15
3	3	6	9	12	15	1	4	7	10	13	16	2	5	8	11	14
4	4	8	12	16	3	7	11	15	2	6	10	14	1	5	9	13
5	5	10	15	3	8	13	1	6	11	16	4	9	14	2	7	12
6	6	12	1	7	13	2	8	14	3	9	15	4	10	16	5	11
7	7	14	4	11	1	8	15	5	12	2	9	16	6	13	3	10
8	8	16	7	15	6	14	5	13	4	12	3	11	2	10	1	9
9	9	1	10	2	11	3	12	4	13	5	14	6	15	7	16	8
10	10	3	13	6	16	9	2	12	5	15	8	1	11	4	14	7
11	11	5	16	10	4	15	9	3	14	8	2	13	7	1	12	6
12	12	7	2	14	9	4	16	11	6	1	13	8	3	15	10	5
13	13	9	5	1	14	10	6	2	15	11	7	3	16	12	8	4
14	14	11	8	5	2	16	13	10	7	4	1	15	12	9	6	3
15	15	13	11	9	7	5	3	1	16	14	12	10	8	6	4	2
16	*16*	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

TABLE II-2: Unmodified Modulo Product **Operation: p=19 (prime), m=4 bits**

— KEY VALUES →

INPUT VALUES ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	*16*
2	2	4	6	8	10	12	14	*16*	*18*	1	3	5	7	9	11	13
3	3	6	9	12	15	*18*	2	5	8	11	14	*17*	1	4	7	10
4	4	8	12	*16*	1	5	9	13	*17*	2	6	10	14	*18*	3	7
5	5	10	15	1	6	11	*16*	2	7	12	*17*	3	8	13	*18*	4
6	6	12	*18*	5	11	*17*	4	10	*16*	3	9	15	2	8	14	1
7	7	14	2	9	*16*	4	11	*18*	6	13	1	8	15	3	10	*17*
8	8	*16*	5	13	2	10	*18*	7	15	4	12	1	9	*17*	6	14
9	9	*18*	8	*17*	7	*16*	6	15	5	14	4	13	3	12	2	11
10	10	1	11	2	12	3	13	4	14	5	15	6	*16*	7	*17*	8
11	11	3	14	6	*17*	9	1	12	4	15	7	*18*	10	2	13	5
12	12	5	*17*	10	3	15	8	1	13	6	*18*	11	4	*16*	9	2
13	13	7	1	14	8	2	15	9	3	*16*	10	4	*17*	11	5	*18*
14	14	9	4	*18*	13	8	3	*17*	12	7	2	*16*	11	6	1	15
15	15	11	7	3	*18*	14	10	6	2	*17*	13	9	5	1	*16*	12
16	*16*	13	10	7	4	1	*17*	14	11	8	5	2	*18*	15	12	9

TABLE II-3: Key values and holes associated with them
Each row contains input values producing a given output, except for holes (black squares).
Each column contains key values produc ing outputs for a given input, except for holes.

— KEY VALUES →

OUTPUT VALUES ↓	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	10	13	5	4	16	11	12	■	2	7	8	3	15	14	6
2	2	1	7	10	8	13	3	5	15	4	14	16	6	11	9	12
3	3	11	1	15	12	10	14	■	13	6	2	5	9	7	4	■
4	4	2	14	1	16	7	6	10	11	8	9	13	12	3	■	5
5	5	12	8	6	1	4	■	3	9	10	16	2	15	■	13	11
6	6	3	2	11	5	1	9	15	7	12	4	10	■	14	8	■
7	7	13	15	16	9	■	1	8	5	14	11	■	2	10	3	4
8	8	4	9	2	13	14	12	1	3	16	■	7	5	6	■	10
9	9	14	3	7	■	11	4	13	1	■	6	15	8	2	12	16
10	10	5	16	12	2	8	15	6	■	1	13	4	11	■	7	3
11	11	15	10	■	6	5	7	■	16	3	1	12	14	13	2	9
12	12	6	4	3	10	2	■	11	14	5	8	1	■	9	16	15
13	13	16	■	8	14	■	10	4	12	7	15	9	1	5	11	2
14	14	7	11	13	■	15	2	16	10	9	3	■	4	1	6	8
15	15	■	5	■	3	12	13	9	8	11	10	6	7	16	1	14
16	16	8	■	4	7	9	5	2	6	13	■	14	10	12	15	1

A "hole" is an output value that will not occur for any input value in the set {1,2,...2^m} of possible input values, given a particular Key value in that set.

Unmodified Modulo

TABLE III-1: Product **Operation: p=37 (prime), m=5 bits**

— KEY VALUE →

INPUT VALUE ↓

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	*32*
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	*32*	*34*	*36*	1	3	5	7	9	11	13	15	17	19	21	23	25	27
3	3	6	9	12	15	18	21	24	27	30	*33*	*36*	2	5	8	11	14	17	20	23	26	29	*32*	*35*	1	4	7	10	13	16	19	22
4	4	8	12	16	20	24	28	*32*	*36*	3	7	11	15	19	23	27	31	*35*	2	6	10	14	18	22	26	30	*34*	1	5	9	13	17
5	5	10	15	20	25	30	*35*	3	8	13	18	23	28	*33*	1	6	11	16	21	26	31	*36*	4	9	14	19	24	29	*34*	2	7	12
6	6	12	18	24	30	*36*	5	11	17	23	29	*35*	4	10	16	22	28	*34*	3	9	15	21	27	*33*	2	8	14	20	26	*32*	1	7
7	7	14	21	28	*35*	5	12	19	26	*33*	3	10	17	24	31	1	8	15	22	29	*36*	6	13	20	27	*34*	4	11	18	25	*32*	2
8	8	16	24	*32*	3	11	19	27	*35*	6	14	22	30	1	9	17	25	*33*	4	12	20	28	*36*	7	15	23	31	2	10	18	26	*34*
9	9	18	27	*36*	8	17	26	*35*	7	16	25	*34*	6	15	24	*33*	5	14	23	*32*	4	13	22	31	3	12	21	30	2	11	20	29
10	10	20	30	3	13	23	*33*	6	16	26	*36*	9	19	29	2	12	22	*32*	5	15	25	*35*	8	18	28	1	11	21	31	4	14	24
11	11	22	*33*	7	18	29	3	14	25	*36*	10	21	*32*	6	17	28	2	13	24	*35*	9	20	31	5	16	27	1	12	23	*34*	8	19
12	12	24	*36*	11	23	*35*	10	22	*34*	9	21	*33*	8	20	*32*	7	19	31	6	18	30	5	17	29	4	16	28	3	15	27	2	14
13	13	26	2	15	28	4	17	30	6	19	*32*	8	21	*34*	10	23	*36*	12	25	1	14	27	3	16	29	5	18	31	7	20	*33*	9
14	14	28	5	19	*33*	10	24	1	15	29	6	20	*34*	11	25	2	16	30	7	21	*35*	12	26	3	17	31	8	22	*36*	13	27	4
15	15	30	8	23	1	16	31	9	24	2	17	*32*	10	25	3	18	*33*	11	26	4	19	*34*	12	27	5	20	*35*	13	28	6	21	*36*
16	16	*32*	11	27	6	22	1	17	*33*	12	28	7	23	2	18	*34*	13	29	8	24	3	19	*35*	14	30	9	25	4	20	*36*	15	31
17	17	*34*	14	31	11	28	8	25	5	22	2	19	*36*	16	*33*	13	30	10	27	7	24	4	21	1	18	*35*	15	*32*	12	29	9	26
18	18	*36*	17	*35*	16	*34*	15	*33*	14	*32*	13	31	12	30	11	29	10	28	9	27	8	26	7	25	6	24	5	23	4	22	3	21
19	19	1	20	2	21	3	22	4	23	5	24	6	25	7	26	8	27	9	28	10	29	11	30	12	31	13	*32*	14	*33*	15	*34*	16
20	20	3	23	6	26	9	29	12	*32*	15	*35*	18	1	21	4	24	7	27	10	30	13	*33*	16	*36*	19	2	22	5	25	8	28	11
21	21	5	26	10	31	15	*36*	20	4	25	9	30	14	*35*	19	3	24	8	29	13	*34*	18	2	23	7	28	12	*33*	17	1	22	6
22	22	7	29	14	*36*	21	6	28	13	*35*	20	5	27	12	*34*	19	4	26	11	*33*	18	3	25	10	*32*	17	2	24	9	31	16	1
23	23	9	*32*	18	4	27	13	*36*	22	8	31	17	3	26	12	*35*	21	7	30	16	2	25	11	*34*	20	6	29	15	1	24	10	*33*
24	24	11	*35*	22	9	*33*	20	7	31	18	5	29	16	3	27	14	1	25	12	*36*	23	10	*34*	21	8	*32*	19	6	30	17	4	28
25	25	13	1	26	14	2	27	15	3	28	16	4	29	17	5	30	18	6	31	19	7	*32*	20	8	*33*	21	9	*34*	22	10	*35*	23
26	26	15	4	30	19	8	*34*	23	12	1	27	16	5	31	20	9	*35*	24	13	2	28	17	6	*32*	21	10	*36*	25	14	3	29	18
27	27	17	7	*34*	24	14	4	31	21	11	1	28	18	8	*35*	25	15	5	*32*	22	12	2	29	19	9	*36*	26	16	6	*33*	23	13
28	28	19	10	1	29	20	11	2	30	21	12	3	31	22	13	4	*32*	23	14	5	*33*	24	15	6	*34*	25	16	7	*35*	26	17	8
29	29	21	13	5	*34*	26	18	10	2	31	23	15	7	*36*	28	20	12	4	*33*	25	17	9	1	30	22	14	6	*35*	27	19	11	3
30	30	23	16	9	2	*32*	25	18	11	4	*34*	27	20	13	6	*36*	29	22	15	8	1	31	24	17	10	3	*33*	26	19	12	5	*35*
31	31	25	19	13	7	1	*32*	26	20	14	8	2	*33*	27	21	15	9	3	*34*	28	22	16	10	4	*35*	29	23	17	11	5	*36*	30
32	*32*	27	22	17	12	7	2	*34*	29	24	19	14	9	4	*36*	31	26	21	16	11	6	1	*33*	28	23	18	13	8	3	*35*	30	25

Example of holes: 29, 31 do not occur given K=2

Overflowing values (>32), which would be remapped to holes with pseudogroup operation, are shown in italics.

TABLE III-2: Key values and holes associated with them
Each row contains keys values producing a given output, except for holes (black squares).
Each column contains key values producting outputs for a given input, except for holes.

— KEY VALUES →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	19	25	28	15	31	16	14		26	27		20	8	5	7	24		2	13	30	32	29	17	3	10	11	4	23	21	6	22
2	2	1	13	19	30	25	32	28	29	15	17	31	3	16	10	14	11		4	26	23	27	21		6	20	22	8	9	5	12	7
3	3	20	1	10	8	19	11	5	25	4	7	28	23	24	15	21		31	6	2	16	22	13	14	9	30		12	32	26	18	29
4	4	2	26	1	23	13	27	19	21	30		25	6	32	20	28	22	29	8	15	9	17	5	31	12	3	7	16	18	10	24	14
5	5	21	14	29	1	7	6		17	19	24	22	26	3	25		9	27	10	28	2	12		11	15	13	18	20	4	31	30	
6	6	3	2	20	16	1	22	10	13	8	14	19	9	11	30	5		25	12	4	32	7	26	28	18	23	29	24	27	15		21
7	7	22	27	11	31	32	1	24	9		4	16	29	19		12	20	23	14	17	25	2	18	8	21		3	28	13		5	6
8	8	4	15	2	9	26	17	1	5	23	31	13	12	27	3	19	7	21	16	30	18		10	25	24	6	14	32		20	11	28
9	9	23	3	30	24	20		15	1	12	21	10	32		8	26	31	19	18	6	11	29	2	5	27	16	25		22	4	17	13
10	10	5	28	21	2	14	12	29		1	11	7	15	6	13		18	17	20	19	4	24	31	22	30	26		3	8	25	23	
11	11	24	16	12	17	8	28	6	30	27	1	4		14	18	3	5	15	22	32		19	23	2			10	7	31	9	29	20
12	12	6	4	3	32	2	7	20	26	16	28	1	18	22	23	10	29	13	24	8	27	14	15	19		9	21	11	17	30		5
13	13	25	29	31	10		23		22	5	18		1	30	28	17	16	11	26	21	20	9	7		2	19	32	15	3	14	4	27
14	14	7	17	22	25	27	2	11	18	31	8	32	21	1		24	3	9	28		13	4		16	5	29	6	19	26		10	12
15	15	26	5	13	3	21	18	25	14	20		29	4	9	1	31	27	7	30	10	6		28		8	2	17	23	12	19	16	
16	16	8	30	4	18	15		2	10	9	25	26	24	17	6	1	14	5	32	23		31	20	13	11	12	28	27		3	22	19
17	17	27	18	32		9	13	16	6		15	23	7	25	11	8	1	3			29	26	12	30	14	22	2	31	21	24	28	4
18	18	9	6	23	11	3	29	30	2	24	5	20	27		16	15	25	1		12	22	21	4	10	17	32	13		7	8		26
19	19	28	31	14	26		8	7		13	32	17	10	4	21	22	12		1	25	15	16		27	20	5	24	2	30	29	3	11
20	20	10	19	5	4	28	24	21	31	2	22	14	30	12	26	29			3	1	8	11	25	7	23	15		6	16	13	9	
21	21	29	7		19	22	3		27	28	12	11	13	20	31		23	32	5	14	1	6	17	24	26	25	9	10	2		15	18
22	22	11	32	24		16	19	12	23	17	2	8		28		6	10	30	7	27	31	1	9	4	29		20	14	25	18	21	3
23	23	30	20	15	12	10		26	19	6	29	5	16		4	13		28	9	3	24		1	21	32	8	31	18	11	2	27	25
24	24	12	8	6	27	4	14	3	15	32	19	2		7	9	20	21	26	11	16	17	28	30	1		18	5	22		23		10
25	25	31			5		30	17	11	21	9		19	15	14	27	8	24	13	29	10	23	22	18	1	28	16	26	20	7	2	32
26	26	13	21	25	20	29	9	31	7	10			2	23	19		32	22	15	5	3	18	14		4	1	27	30	6	28	8	17
27	27	32	9	16		23	25	8	3		26	30	22	31	24	4	19	20	17	18		13	6	15	7	11	1		29	12	14	2
28	28	14		7	13	17	4	22		25	16	27	5	2	29	11	6	18	19	31	26	8		32	10	21	12	1	15		20	24
29	29		22		28	11	20		32	14	6	24	25	10		18	30	16	21	7	19	3	27	12	13	31	23	5	1	17	26	9
30	30	15	10	26	6	5		13	28	3		21	8	18	2	25	17	14	23	20	12		19	29	16	4		9	24	1	32	31
31	31			17	21		15	27	24	29	23	18	28	26	7	32	4	12	25		5	30	11	9	19	14	8	13	10	22	1	16
32	32	16	23	8		30	31	4	20	18	13	15	11		12	2	28	10	27	9		25	3	26	22	24	19	17		6	7	1

| INPUT VALUES ↓

A "hole" is an output value that will not occur for any input value in the set {1,2,...2^m} of possible input values, given a particular Key value in that set.

Encryption (forward)

Sub-Block size

a = plaintext $\qquad x$ = key $\qquad M = 2^N$

1. $z = ax \bmod p$ $\qquad p$ = prime just above M

2. If $z > M$,

2.1 $\qquad k = z - M$

2.2 $\qquad z = kx - (p - M - 1) \quad \bmod p$

end.

Decryption (inverse)

z = ciphertext $\qquad x^{-1}$ = inverse key

1. $\qquad k = x^{-1}(z + (p - M - 1)) \bmod p$

2. If $1 \le k \le (p - M - 1)$, 2.1 $z = M + k$; end

2.2 $\qquad a = z x^{-1} \bmod p$

64-000000010000000100000001000000010000. /00001000000000000000000000000
63-00000001000000010000000100000001000000010000000100000000000000000
62-00000000100000000100000000100000000100000000100000000100000000000
61-00000000001000000000100000000010000000001000000000100000000010000
60-00000000000100000000000100000000000100000000000100000000000100000000
59-10000000000001000000000000100000000000010000000000001000000000000100000
58-0000001000000100000000000001000000000000010000000000000100000000000000
57-000000000000011000000000000001000000000000001000000000000001000000000
56-00010000000000010001000000000001000000000000000100000000000000010000
55-0000000001000000100000000100000001000000000000000100000000000000001
54-000000000000001001000000000000001001000000000000000010000000000000
53-100000000000000000110000000000000000001100000000000000000100000000000
52-000010000000000000100001000000000000010000100000000000001000000
51-00000000100000000001000000000100000000001000000001000000000001000
50-000000000000010000000100000000000001000000001000000000000010000000010
49-000000000000000001000010000000000000000010000100000000000000001000
48-000000000000000000000101000000000000000000000101000000000000000000
47-110000000000000000000000011000000000000000000000000110000000000000
46-000100010000000000000000010001000000000000000000000010001000000000
45-000000100000010000000000010000000100000000000000000001000000100000
44-000000000100000000010000000100000000010000000000000000010000000001
43-000000000000100000000000001010000000000001000000000000001000000000
42-001000000000000100000000000001001000000000001000000000000001000000
41-000000010000000000100000000000100000000100000000001000000000001000
40-000000000000100000000100000000100000000000001000000001000000000100
39-000000000000000000010000001000000100000000000000000010000001000000001
38-000000000000000000000000100001000010000000000000000000000100001000
37-000000000000000000000000000001001001000000000000000000000000000100
36-000000000000000000000000000000000001110000000000000000000000000000
35-111000000000000000000000000000000000000111000000000000000000000000
34-001001001000000000000000000000000000001001001000000000000000000000
33-000010000100001000000000000000000000000100001000010000000000000000

— KEY VALUES →

(71 − 65 = 6) ← 2^N+1 is special case where

$\Sigma k = 0$

12 =

32-000000100000010000001000000000000000000000100000010000001000000000000
31-000000000100000000100000000100000000000000000100000000100000000100000
30-000000000001000000000001000000000001000000000001000000000001000000010
29-000000000000010000000000000100000000000000100000000000000100000000000
28-010000000000000010000000000000010000000000000010000000000000010000000
27-000000100000000000001000000000000000000100000000000000100000000001000
26-000000000000010000000100000000000000000000000100000000000000000100001
25-000000000000000000001000100000000000000000000000001000100000000000010
24-000000000000000000000000010000000000000000000000000000010000000000000
23-010100000000000000000000000010100000000000000000000000000101000000000
22-000010000010000000000000000000000010000010000000000000000000100000100
21-000000010000000100000000000000000100000001000000000000000100000001000
20-000000000010000000000001000000000100000000001000000000000100000000100
19-000000000000010000000000000000100000000000010000000000000000100000000
18-000000000000000100000000000000000000010000000000000000000000000100000
17-001000000000000000001000000000000000000100100000000000000000100100000
16-000000100000000000000001000000000000000001000000000000000000001000000
15-000000000001000000000000000100000000000000000010000000000000000010000
14-100000000000000100000000000000000010000000000000000000100000000001000
13-000001000000000000000100000000000000000100000000000000001000000000001
12-000000000010000000000000000001000000000000000001000000000000000100000
11-000010000000000001000000000000000000100000000000000000001000000000100
10-000000000001000000000100000000000000001000000000000000010000000000010
9-000000000000000000100000000001000000000100000000010000000001000000000100
8-000000000000000000000000100000000100000000100000000100000000100000000010
7-00000000000000000000000000000000001000000010000000100000001000000010000001
6-000000000000000000000000000000000000001000001000001000001000001000000
5-00000000000000000000000000000000000000000000010000100001000010000100
4-000000000000000000000000000000000000000000000000000100010001000100010
3-0000000000000000000000000000000000000000000000000000000001001001001000
2-000000000000000000000000000000000000000000000000000000000000000101010
1-00000000000000000000000000000000000000000000000000000000000000000000

— KEY VALUES →

— OUTPUT HOLE VALUES →

k_1 k_2 k_3 k_4 k_5 k_6

$p = 71$
$m = 64$

Examples:

$Y_1 = 1\cdot(x-6) = 1\cdot x - 6$

$Y_2 = 2\cdot(x-3) = 2\cdot x - 6$

$Y_3 = 3\cdot(x-2) = 3\cdot x - 6$

$Y_4 = 4\cdot x - 6$

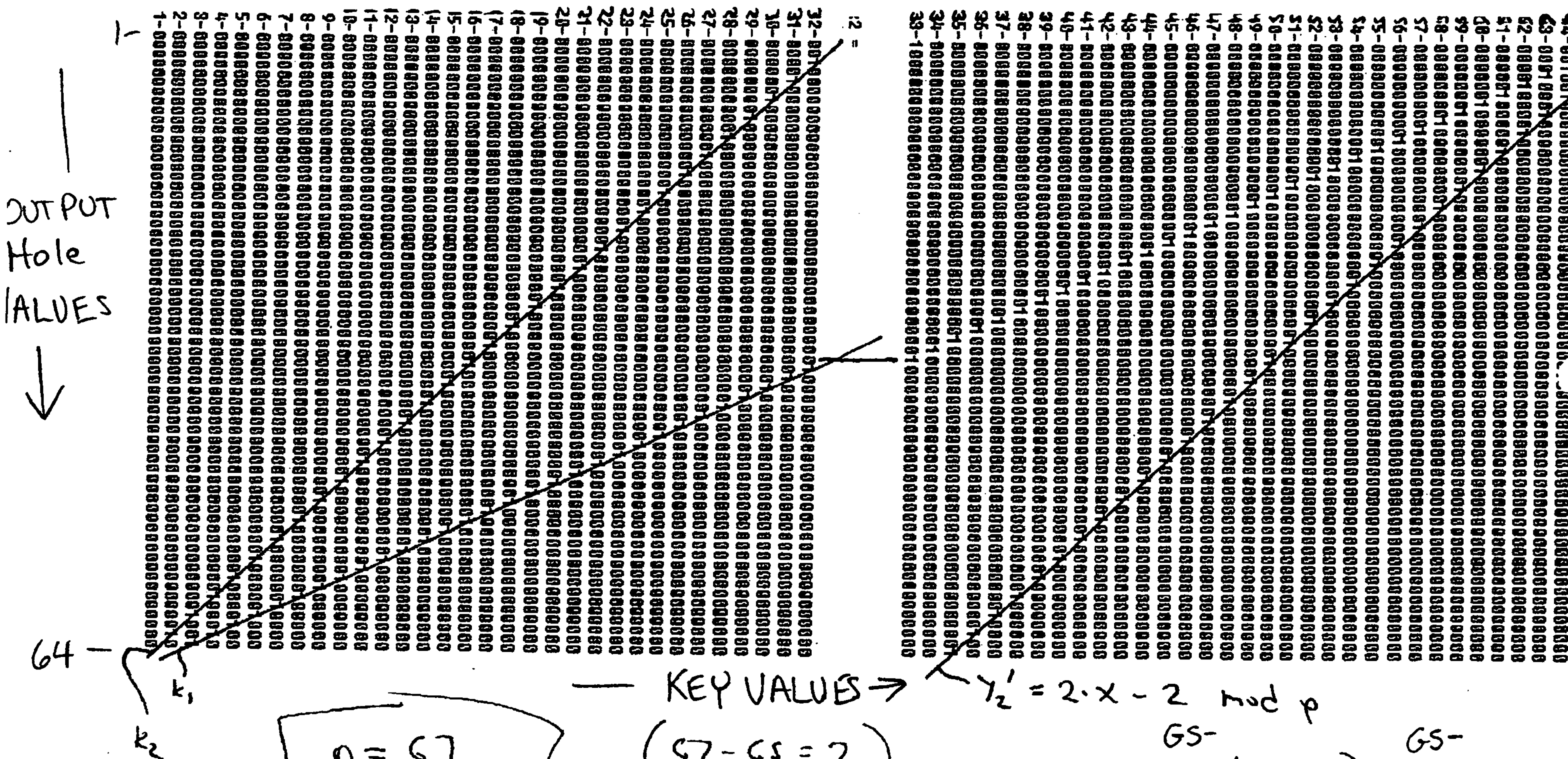
$Y_5 = 5\cdot x - 6$

$Y_6 = 6\cdot x - 6$

$\therefore Y_k = k\cdot x - (p-(m+1))$

$k_1 = 1$
$k_2 = 2$
...
$k_6 = 6$

P-1



— KEY VALUES → $Y_2' = 2 \cdot X - 2 \mod p$

$(67 - 65 = 2)$

$$Y_1 = \overset{GS-}{1} \cdot (X-2) \overset{GS-}{=} 1 \cdot X - 2$$

$$Y_2 = \overset{GS-}{2} \cdot (X-1) \overset{GS-}{=} 2 \cdot X - 2$$

$$\boxed{\begin{aligned} p &= 67 \\ M &= 64 \end{aligned}}$$

$k_1 = 1$
$k_2 = 2$

$Y_{21} = Y_{11} + k_1$
$Y_{22} = Y_{12} + k_2$

A PARTICULARLY USEFUL PSEUDOGROUP EQUATION:

$$Y(x, K) = M + 1 - \left[kx - (p - (M+1)) \right] \mod p$$

IF $Y(x,k) > M$, NO HOLE

P-2

```
64-0000000001 0000000001 0000000001 000000  000000001 0000000001 0000000010
63-0000001 001 0000000001 0000000001 0000000001 0000000001 0000000001 0000
62-0001 0000001 0001 000001 0000000001 000000000001 0000000001 000000000
61-0000000000011 00000000011 00000000001 00000000001 000000000001 0000
60-00001 0000001 00001 0000001 00001 0000001 0000000000001 0000000000
59-000000000001 001 00000000001 001 00000000001 001 0000000000001 00000000
58-010000000000001 01 00000000000001 01 000000000000001 01 000000000001 0000
57-000001 000000001 0000001 000000001 0000001 000000001 0000001 000000001
56-0000000000001 00001 0000000000001 00001 0000000000001 00001 00000000000010
55-000000000000000011 0000000000000000011 0000000000000000011 0000000000
54-001001 00000000000001 001 0000000000000001 001 000000000000000001 001 0000
53-0000001 000001 000001 000001 0000000000001 000001 00000000000001 0000
52-1 000000001 000000001 1 00000000001 00000001 000000000001 000000000010
51-0000001 000000 1 000001 0000001 0000001 000001 000000000000001 00000
50-00000000000001 0001 0001 0000000000000001 0001 0001 00000000000000000
49-00000000000000000000111 0000000000000000000000111 0000000000000000
48-01 01 00000000000000000001 01 01 000000000000000000001 01 01 000000000
47-00001 00001 00000000000000001 00001 00001 000000000000000001 00001 000001 00
46-0000001 0000001 0000000000001 0000001 0000001 000000000001 0000001 00
45-00001 000001 0000000000001 000001 000000000001 00000000001 000001 00000000
44-0000000000011 0000000000000011 00000000000001 0000000000000011 000000
43-0001 0000000000001 0001 00000000001 0001 0000000000001 000000000000001 0001
42-000000001 000000000001 0000000001 01 00000001 0000000001 000000000100
41-00001 0000000001 000000001 000000001 00001 000000001 000000001 00000001
40-00000000001 0000001 000001 000001 0000000000001 000001 0000001 00000
39-000000000000000001 0001 00001 00001 0000000000000000000001 00001 000010
38-0000000000000000000000000001 001 001 001 00000000000000000000000000001 001
37-0000000000000000000000000000000001111 0000000000000000000000000000000
36-1111 000000000000000000000000000000001111 00000000000000000000000000
35-001001 001001 00000000000000000000000001 001 001 001 00000000000000000
34-00001 00001 00001 00001 00000000000000001 00001 00001 00001 000000000
33-0000001 000001 000001 000001 0000000000001 000001 000001 000001 000
```

K2 =

```
32-00000001 00000001 00000001 00000001 00001 00000001 00000001 00000
31-01 000000001 00000001 00000000001 0000001 01 0000000001 000000001
30-0000000 01 0001 000000000001 000000000001 0001 000000001 0001 0000000
29-1 000000000000011 00000000000001 0000000000000011 00000000000011 0000
28-000001 000000000 1 000001 00000000001 00000000001 000001 000000000100
27-001 00000001 0000001 000000000001 0000001 0000001 00000000001 000000
26-000000000001 00001 0001 0000000000000001 00001 0001 000000000000010
25-0000000000000000001 01 01 000000000000000000001 01 01 0000000000000000
24-11 0000000000000000000001 1 1 000000000000000000000111 000000000000
23-0001 0001 000000000000000001 0001 0001 000000000000001 0001 0001 000000
22-0000001 000001 000000000000001 000001 000001 000001 000001 000000
21-00000000001 0000000001 00000000001 00000000001 000000001 1 000000000100
20-000001 000001 000000000001 000001 000000000001 000001 000001 00000
19-00000000000001 001 000000000000001 001 00000000000000001 001 0000000000
18-11 0000000000000000011 0000000000000000011 0000000000000000011 000000000
17-00001 00001 0000000000001 00001 0000000000001 00001 00000000000001 00001 000
16-00000001 00000001 0000001 0000001 000001 000000001 000001 0000000
15-000000000001 0000000000001 01 00000000000001 01 0000000000001 01 000000
14-001 000000000000001 00000000000001 001 000000000001 001 00000000000001 00100
13-0000001 00000000000001 00000000000001 00000001 00001 0000001 00001 0000
12-1 0000000000001 000000000001 00000000001 0000000000011 0000000001 100
11-000001 0000000001 00000000001 000000001 0000000001 000001 000100
10-001 000000000001 000000001 000000001 000000001 00000001 0000000010
9-1 00000001 00000001 0000001 00000001 00000001 000000001 000000001
8-0000001 000001 000001 000001 0000001 000001 000001 000000
7-000000000000001 000001 000001 000001 000001 000001 000001 00000
6-00000000000000000000000001 000001 00001 00001 00001 000001 000001 000
5-000000000000000000000000000000001 00001 00001 00001 00001 00001 00010
4-00000000000000000000000000000000000001 0001 0001 0001 0001 0001 000
3-000000000000000000000000000000000000000000001 001 001 001 001 001
2-0000000000000000000000000000000000000000000000000000000001 01 01 010
1-00000000000000000000000000000000000000000000000000000000000000000000
```

$p = 73$

$M = 64$

$73 - 65 = 8$

For $x = 51$, $k = 7$

$y = \cancel{8} 7 \cdot 51 - 8 \bmod 73$

$= 65$ (no hole)

$\therefore y_8 = 8 \cdot x - 8$

$y_8(x = 3) = 16$

$(3, 16)$ should be a hole - it is!

$y_1 = 1 \cdot x - 8$

$y_2 = 2 \cdot x - 8$

$y_3 = 3 \cdot x - 8$

$x = 52, k = 7$

$= 64$

P-3

```
function h = holes(y,p,M,short);

% h = holes(y,p,N, short);
% CONFIDENTIAL AND PROPRIETARY
% Edwin A. Suominen

% Finds "holes" - skipped values of set {0,1}^N in result
% of x*y mod p.

% Number of values in set S:{0,1}^N
% M = 2^N;

% For vector inputs...
for k=1:length(y)

s = 1:M;    % Working array of values in set S

% Zero out values in set that occur ("non-holes")
  for i = 1:M
    j = product(i,y(k),p);  % xy mod p
    % Zero out if not a hole
    if j<=M, s(j) = 0; end
  end

% Sort decending to get holes first
  z = -sort(-s);

if nargin>3,
 % There can be no more than p-2^N-1 holes.
 % Limit size of result vector(s) accordingly.
  z = z(1:p-M-1);
% Add result vector to array (if vector y)
  h(:,k) = z';
else
  h(:,k) = s';
end


end

end
```

```
function [y1,y2] = holeplot(p,M);

% y = holeplot(p,M)

% Modulus is 2^N+k, where k is odd
% p = M + k;

% Create 1010...matrix of holes
for i = 1:M/2,
  % Get holes for each column
  s = holes(M+1-i,p,M);
  % Convert to 1010... format
  s = s>0;
  % Convert to text string
  col = M+1-i
  if col<10,
    rl = [' ' num2str(col,2)];
  else
    rl = num2str(col,2);
  end
  y1(i,:) = [ rl '-' num2str(s,1)];
end

for i = 1:M/2,
  % Get holes for each column
  s = holes(M+1-(i+M/2),p,M);
  % Convert to 1010... format
  s = s>0;
  % Convert to text string
  col = M/2+1-i
  if col<10,
    rl = [' ' num2str(col,2)];
  else
    rl = num2str(col,2);
  end
  y2(i,:) = [ rl '-' num2str(s,1)];
end
```